SINCLAIR QL WORLD

The Premier Magazine for Sinclair QL Users

4 TO 7 10/11 16 18 22
26 32 36 DISC DRIVES
40 41 49

EVERY MONTH
£1.75
SEPTEMBER 1990

SUPERBASIC
Taking Minerva into account

Getting Organised on the QL

TIPS ON DISKS

Software file
CHINESE CHESS

New:
PROGRAMMING IN C

TROUBLESHOOTER

DIY TOOLKIT:
NEWCHAN%, LOOKUP%, UPPER%, LOWER%

ISSN 0951-9335
09
9 770951 933009

SINCLAIR QL WORLD

Editor
Helen Armstrong

Production Controller
Jayne Penfold

Designer
Jeff Gurney

Advertising Sales
Jason Newman

Magazine Services
Sheila Baker

Advertising Production
Michelle Evans

Group Advertising Manager
Richard Vaughan

Group Editor
John Taylor

Publishing Director
Ray Lewis

Managing Director
Peter Welham

Sinclair QL World
Panini House
116-120 Goswell Road
London EC1V 7QD
Telephone 071-490 7161
ISSN 026806X

Unfortunately, we are no longer able to answer enquiries made by telephone. If you have any comments or difficulties, please write to The Editor, Open Channel, Trouble Shooter, or Psion Solutions. We will do our best to deal with your problem in the magazine, thuough we cannot guarantee individual replies.
Overseas rates on request.
Published by Maxwell Consumer Magazines, A Division of MCPC Ltd. S M Distribution, Streatham, London SW1. 01 677 8111.
Subscription information from: MCM Subscription Dept, Lazahold Ltd., PO Box 10, Roper St, Pallion Ind. Est., Sunderland SR4 4SN. £21.00 U.K., £24.70 Europe, Middle East £25.80, Far East £27.60, Rest of World £26.20, U.S.A. $45.00. Airmail rates available on request.
Typesetting by Adtec Typographics, Britannia Court, Basildon, Essex. Tel: (0268) 591110.
Printing by Cradley Print.
Sinclair QL World is published on the fourth Wednesday preceding cover date.

CONTENTS

SEPTEMBER 1990

NEXT MONTH

ARCHIVE POWER
The final part of our four-part series

INTEGERS AND FLOATING POINTS
How the QL represents numbers internally

QL SCENE

Data Design from Belgium

Progs, the European software house run by Joachim and Nathan Van der Auwera, is releasing a new database program, *Datadesign*. The program is designed among other things to handle multiple lines in a field, without a set field length.

The all machine-code Datadesign uses a pointer and menu-extension environment for ease of use, multiuse, multitasks, sorts on two levels and also by number and alphanumeric order, has a fast-find option, and finds by record or by field. The program will view, jump to a record, delete a batch of records or print a selection of records.

The makers describe it as having a printer-independent print-driver and disk-independent for speed.There is a Basic routine to transfer Archive export files to Data design.

Datadesign is available only on 3.5in. disk, and a memory expansion (they do not specify how much) is necessary. Datadesign costs £60 sterling or 3000 Belgian Francs, the latter which is preferred, post paid, from **Progs (van Der Auwera), Haachtstraat 92, 3008 Veltem, Belgium, Local phone no.016/48 89 52.**

Euro Show

Club Sinclair BruQsL of Brussels are once again organising a European Microfair 1990, subtitled The Big Sinclair Show, on 6 October at the Eurovolleycenter in Vilvoorde, Brussels where it was such a success in 1989. This year, the Show is taking the whole of the sports hall to gain more space.

Organiser Jacques Tasset requests that anybody who is interested in taking a stand or giving a demonstration should contact him at the following address:

Jacques Tasset, Club Sinclair BruQsL, Aarlenstratt 104, B-1040 Brussels, Belgium. The local phone number is 02/233 12 22.

Erratum

CGH Services reports that the reference to issue 3 of *QL Leisure Review* in last month's *Troubleshooter* is in error. "*QL Technical Review* has reached number 3 . . . *QL Leisure Review* (the successor to QL Adventurers' Forum) has yet to be published but will be as soon as we get enough material," says Richard Alexander. "Please correct the impression as we are likely to get queries from people wondering where their subscription copies have got to."

Next Fair

The next **All Formats Computer Fair** takes place on 1st and 2nd of September at the new Horticultural Halls, Westminster, London, closely following the previous Fair on 4th and 5th August.

Organiser Bruce Everiss is advertising widely in European newspapers and computer magazines, to encourage visitors from Europe to visit the fair looking for bargain software and peripherals. Complimentary tickets are being sent to user groups and clubs.

In Spain

CUQ (Circulo Usuarios QL) started as a sub-group of QLAVE, the first Spanish QL User group (see *QL World*, April 1988). It went independent in September 1988.

The club has a software library of hundreds of Spanish QL programs written by members, and "thousands of pages" in their disk magazine CUQ. They also support the Cambridge Computer Z88, and the group's Editor is the author of Z88 Forth, for the Z88, which is also available from the UK Z88 user group.

Write to **Salvador Merino, Paeso Maritimo 86, EDF, Berlin, 29640 Loss Boliches (Malaga), Spain.**

SCREEN SAFETY

The EC Council of Ministers has adopted directives on minimum health and safety requirements for work with vdus (now apparently known officially as "display screen equipment"). The Health and Safety Commission will be considering what proposals should be drawn up to give effect to the Directives in the UK, with an eye on the Health and Safety and Work Act 1974,. Consultive documents are expected in 1991.

The basic requirements of the Directive are:

— that employers should analyse display screen workstations to evalute safety and health conditions. "taking appropriate measures to remedy any risks found".

— employers must ensure that workstations entering service after 31 December 1992 meet the requirements contained in the Annex to the Directive, which sets standards for display screens, keyboard, furniture, lighting, working environment, task design and software.

— employers must ensure that workstations in service before 21 December 1992 are adapted to comply with the requirements in the Annex within four years of that date, and

— provide for workers to receive information and training and for consultation and participation of workers and/or their representatives.

— employers must plan activities so that daily work on a display screen is periodically interrupted by breaks or changes of activity, and

— give workers an entitlement to an eye and eyesight test before starting display screen work, at regular intervals thereafter, and if they experience visual difficulties. Workers will be entitled to an ophthalmological examination if the eye test shows this to be necessary, and they must be provided with special spectacles if these are needed for their work and normal ones cannot be used.

The UK has abstained in the vote on adoption of the display screen equipment Directive because of doubts about the scope of the directive and the eye test requirments.

The doubts on these two points may have been raised by the mention of "software" in the second point above, and by the mention of "special spectacles" in the final point.

Users who suffer from visual stress and headaches while using vdus have often blamed bad screen colouring and contrast, excessive glare and uncomfortable working angles.

These factors are fundamental by-products of computer and workstation design, and not primarily suitable for treatment by changes in software design, or by special eyeware. These suggestions seem to shift the focus away from the real problem, to "fixing" the software or the user instead.

Since office seating design has improved and manual typewriters have gone, the complaints of keyboard-operators have shifted away from back trouble to vdu strain and RIS (repetitive strain injury, similar to tennis elbow).

OPEN CHANNEL

Open Channel is where you have the opportunity to voice your opinions in *Sinclair QL World*. Whether you want to ask for help with a technical problem, provide somebody with the answer, or just sound off about something which bothers you, write to: Open Channel, Sinclair QL World, 116/120 Goswell Road, London EC1V 7QD.

Tree Dump

I am trying to find out if anybody has incorporated a dump to printer at various stages of the program *Family Tree* which I bought from the microdrive exchange. Obviously it is possible to print out data files but it would be of great value to be able to print out actual family displays.

J M Grant
Bearsden
Glasgow

Editor's comment: We have sent Mr Grant a copy of the original article, but if anybody has experience of doing this with Family Tree please write and we will put them in touch. Unfortunately author Andy Carmichael is one of only two MDX authors that we have lost contact with over the years. The other is John Banks, should anybody find him.

Words

I want to say a big thank you for the many interesting articles in the *QL World*, and also to the writers of software, and the software and hardware firms and dealers for making the QL work better, faster and more reliable.

I have been working on a program of my own containing over 120,000 words and most of their meanings, synonyms or definitions. It is ideal as a crossword-puzzle solver. I have put these words into *QSpell* via Quill and have a 120,000-word-plus dictionary. (You go for a three-hour walk with your wife while the program lists it to screen and tells you the total word-count). I am also working on a PC version.

There have been some articles in *QL World* for beginners, what I find lacking are articles for gradually understanding the harder stuff. Even books on the subject make the harder stuff even harder to understand

To be specific, "what I like and understand in QL World", I understand: most off the adds, most *Open Channel*, *most Technical Helpline*, most Progs. I find hard to understand (or can't): *SuperBasic*. I could just about understand *Better Basic* of 1987, keep to the 1987 style please Mike Lloyd. *DIY Toolkit* is hardest to understand. And there are all sorts of things about printers I don't understand. Thank you again for a wonderful magazine.

Paul Merdinian
Ilminster
Somerset

PS Most of this document has been through my *Spellbound Special*.

Editor's comment: This is certainly a case in favour of spell checkers; it did leave one deliberate mistake, which I have left in place for the sharp-eyed.

I would have thought that Mike Lloyd's SuperBasic is a prime example of an article about gradually understanding the harder stuff. If you worked through Better Basic a few times until you get the hang of it, you should be able to step up.

Editor's notebook

A new cover design this month, continuing the theme of the many interlocking aspects of the QL, which sometimes fit neatly together, and at other times seem to be in search of the Missing Link. Another new piece this month is the start of a series on programming in C, which is aimed at people with a modest experience of programming, but can be read and appreciated by beginners as well. Bryan Davies goes into some depth on the characteristics of different classes of disk, which the new user can keep near at hand to answer some of those basic but irritating questions that crop up. And now a couple of items of Stop Press: There is a Quanta Workshop at Rayne Village Hall, Core Road, Rayne, Nr. Braintree, Essex, 1-2 September. Call Bob Gingell 081-508 8370 for information. Secondly, following the European Microfair on 6 October in Brussels (see QL Scene) there will be another international QL show in Northern Italy on 27-28 October.

Blank

I have networked two QLs. One QL has a Trump Card and the other one has a 512K Expanderam with a disk card. Both QLs have a single disk drive. I have experienced the following problems while networking.

On both machines I have installed the screen dimming facility to protect the monitor. If no keys are touched for three minutes, the screen goes blank. This creates problems when I send messages on the network, since the person on the remote QL will not know that a message has come. To get over this, I wish to BEEP the remote QL before sending the message. How can this be done?

Is it possible to view the screen on the remote networked QL?

If the remote QL is running, say, Quill in a Taskmaster environment and I send a message to that QL, the program (Quill in this case) underneath gets corrupted and unusable. This does not happen when working in a Qram Hotkey environment. Does the network message program fail to close its channels? How can this be overcome?

Now a tip to other networkers. I have used the NFS_use command to access the disk drive and ramdisk of the remote QL easily from any program (even Quill). The command is Nfs_use fdk, n3_flpl_n3,ram1_. So fdk1_is flpl_ on the remote QL and fdk2_ is ram1_ on the remote QL, I can continue to use flpl_ as flpl_ on the remote QL. What else can be done on the network?

I use Icicle a lot with all the Psion programs. However, Icicle is not usable in a Qram Hotkey environment. Is there anyone who can change Icicle to work under Qram? Or can someone make a program like Icicle but using the Pointer Environment?

Sanjay Marwah
Marwah Bhavan
114–A Turner Road
Bandra Bombay 40050
India

Years

Thank you for keeping QL World going after the official demise of dynosaurus QLensis

in 1985. The fossil lives on in many hearts.

I read a while ago that certain people had persuaded Psion to continue work on Archive to version 2.38. I do not use it much but I do frequently use Quill and Abacus. Quill is quite satisfactory, especially with the Turboquill addition. However I am appealing to someone with the ability to make some improvements to Abacus.

Like the following:

1. One key entry of data and cursor movement. When a value has been typed, pressing a cursor key should have the effect of putting the value into the cell as well as moving to the new cell. Also pressing the ENTER key should enter the data and move to a new cell either right or down depending on the setting of the Order command. Not having this facility is a cause of much frustration.

2. Automatic determination of whether an entry is a value or text. Formula entry should be the type forced. This could be by use of the = sign. Of course the inverted commas should also be able to force an entry to the text. This arrangement would save much time and error in entry work.

3. The F5-GOTO function is very useful but there is a great need for a key which would advance the cursor into the left most column of the next row. How about shifted F5? If the order setting is by column then the cursor movement could be to the first row of the next column. This would save much time and repeated cursor key presses.

I should be delighted to see a version 3.0 Abacus.

I should mention that I also own a "compatible" with 1.4 meg drive and 40 meg hard disc but usually prefer to use the QL as I can sit back in my armchair with the QL on my lap and type away. I have a 15 ft cable to a monitor/TV with 20 in screen and find it quite usable.

Thomas D Sutton
St. Louis
USA

Editor's comment: Whom are you calling a fossil? Bear in mind that the dinosaurs lasted for 150 million years. By this reckoning, we should have 149,999,994 years left.

Regrettably the only back issues available from us now are those published under the Maxwell Specialist ownership, ie May 1990 onwards.

QL Box

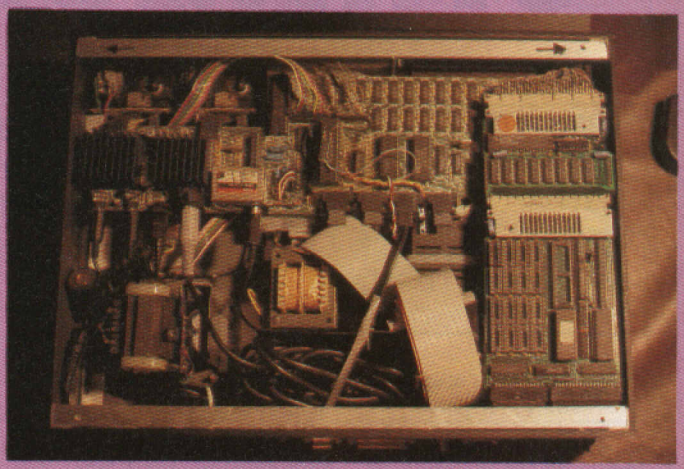
My name is Antonio, I am 29 years old, fond of informatics and a happy QL owner since 1986. Originally I had a Sony MSX machine which I was not satisfied with. One of my friends told me about his QL. I liked the idea of having a big memory and a multi-tasking machine, so having spent (sadly) 850,000 Lire on a computer and tape recorder for the MSX, I sold everything and bought a QL.

I imagine that, having seen the photographs, you will ask yourselves what happened to the original QL. I was not sure I could make the transformation, as I do not know much about electronics and I was told that it was nearly impossible to separate the components of the machine without breaking it.

I decided to transform it to avoid problems with cables of all the bits and pieces. I list what I did even though I didn't know how to do it. Sometimes I tested the machine to see if what I was doing was right, hoping not to break it. I have been lucky; the only thing which does not work any more is mdv_2.

With the help of a friend who had the same problem, we put into a separate box a floppy drive, its power supply, some cables and the QL power supply. The QL Box worked but there was still a problem with the floppy interface which leaned out of the QL, and with the cables which connect it to the QL Box.

The second step was to find a keyboard to substitute for the original one. This was the most complicated thing. I bought the keyboard from an IBM XT-compatible by Keytronic, because there was an electrical failure but the mechanical parts were working. After having rebuilt into it the QL circuit matrix, I connected it. It worked but it was too sensitive. Every time I touched a key cover a lot of letters appeared on the screen. I discovered that the keyboard was capacitive and the QL could not distinguish the first key-press from the following micro-rebounds. I changed the keyboard completely, and now the covers are the only original QL thing on it.



Another problem was separating the floppy interface with its memory expansion from the QL motherboard side connector. I went to the Sandy computer shop which was open at that time and they explained to me that in their opinion it was not possible to extend the cable more than 10cm as signal losses in the cable might be too much.

In the end? It was easy! I wanted to keep the central unit as small as possible, and as the only way was to put it at right angles to the motherboard, I tried the same with the floppy interface and memory expansion, with a cable of at least 15cm, expecting it not to work. But it did.

The last thing to solve was the famous problem of crashing by overheating. This solves itself. It was enough to take out the QL power supply from the original casing and create a good air circulation. There were other difficulties such as finding some of the peripheral components, and it took me two years to finish it.

At the end of 1988 my modified QL was working with this configuration: QL motherboard issue 6, Sandy Super-QBoard with Toolkit 2, Sandy Thruconram 512 KB, Panasonic 3.5in floppy drive giving 720 KB, Hantarex Boxer 12in green monitor, Mannesman-Tally MT80+ 9-in dot matrix printer.

Having planned into the central unit enough space for other bits, I have added a second floppy drive (Citizen slimline 720 KB), with its power supply, and a maintain-clock BClock 1 as well, plus the Spem Digitizer which I have just received. I own a second QL, JS version, which has a BClock 1 with another Hantarex monitor.

That's all. My friends who helped me were Paolo Vanni (the old QL AH and QL Box owner), Sacchi Lionello (ex QL JS-owner, he had to change because of his job), Unberto (great electronics technician, he works on avionics) and I think you should mention Delta Elettronica of Milan, who found for me the necessary peripherals which were so hard to find.

Antonio Gareffa
Milan
Italy

SOFTWARE FILE

"CHINESE CHESS"

INFORMATION:
Program: *Chinese Chess* V4.04
Price: £14.95
Supplier: Ant Publishing, 11 Latton Close, Chilton Didcot, Oxon, OX11 OSU. Tel. (0235)-834254

Bryan Davies moves out for a knight on the tiles with a new QL game — chess with a difference.

Chess in its usual form may frighten off many people. It has an aura of high intelligence and lengthy periods of sedentary head-scratching about it. *Chinese Chess*, as presented by Ant Publishing, uses a board split into two rectangles of 8 × 4 squares each, but it is used as a lattice, with play taking place on the intersections rather than the squares, and there are 90 intersections which the pieces can use. The positions are identified A-I across the board, 0-9 from bottom to top. The 16 pieces of each player are shown as discs marked with Chinese symbols. It is a straightforward game to play, once you have mastered the movements for the pieces, and it is unlikely that the program is so smart that you cannot beat it, sometimes at least. After all, it is not *Psion Chess* (which was written by the man who won the world micro-chess championship).

Before progressing any further with description of the game, it should be pointed out that Ant Publishing have no connection with the supplier that produced the "Ant MS-DOS emulator" that caused some controversy in 1989.

Although the game is said to be a forerunner of modern chess, there are a fair number of differences in the ways pieces can move, a factor which may initially reduce a chess player to beginner level again. The board is split by a "river" across the middle, and there is a "city" at the middle of the rear line on each side. The pieces are General, Counsellor, Horse, Soldier, Cannon, Elephant and Chariot.

The General is roughly equivalent to the King, can move 1 place forwards or sideways, and is confined to the city, an area of 9 positions. Imminent loss of a General means the end of the game. The Counsellor does not seem to have any direct equivalent; it too is confined to the city, and can move 1 place diagonally in any direction. Its role is, at least partially, to protect the General. The Horse is similar to a Knight, able to move 1 place forwards, backwards or sideways, then 1 place diagonally. However, it cannot jump. Its range of movement becomes less restricted as the board gets cleared. The Soldier is equivalent to a Pawn, able to move only 1 place, forwards, until it crosses the river, whereupon it can also move 1 place sideways. In contrast to the Pawn, the Soldier is regarded as being of little value. The Cannon can move any distance forwards, backwards or sideways, but can only capture an opposing piece by jumping over 1 piece (which can belong to either player); all other pieces capture by moving onto an opposing piece's position in the normal chess fashion.

At the beginning of the game, the Cannon tends to initiate the attack, but becomes less useful as the number of pieces reduces. The Elephant moves diagonally, like a Bishop, but only 2 places at a time; it cannot cross the river, so it is basically a defensive piece. The Chariot moves like the Rook/Castle — any distance forwards, backwards or sideways — and is regarded as the most powerful piece once the board becomes sparsely populated. The illustration is a combination of two screens; the top section of the board shows the Chinese-style pieces, whereas they have been replaced (for purposes of the illustration only) by the European-style pieces at the bottom. The menu is displayed at the top right, with the prompt screen for setting-up the board at bottom right (this actually appears only when the "Setup" option has been selected).

Fierce

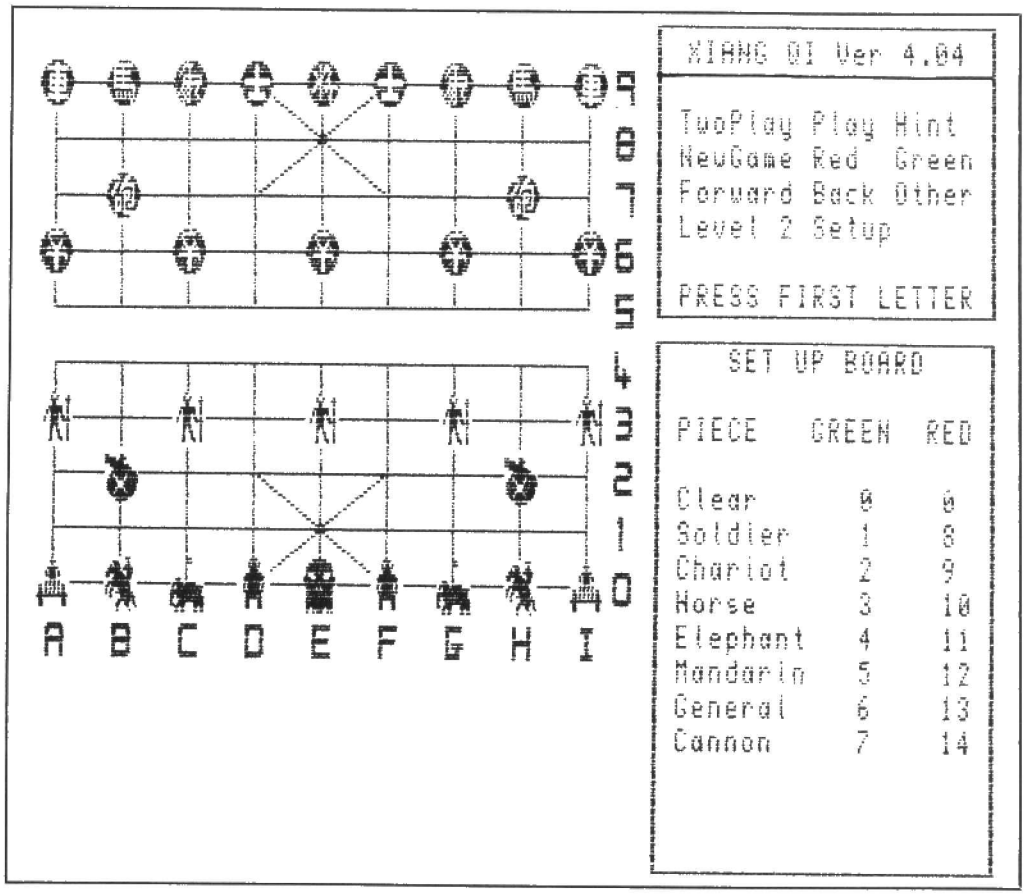
The program came in a wallet reminiscent of those used for the Psion programs, except that a brightly-coloured and fierce-looking Chinese face was on the cover, and the instructions were adorned with Chinese script and a seal. The wallet-sized instruction booklet has 16 pages. It doesn't take long to read, but you have to re-read the details of how the pieces are moved a few times. Details are given of books on Chinese Chess, and there are a couple of sample games (both having an error in them) dating back to 1893.

The practical details of making a backup copy and loading the program are quite straightforward. You can use the supplied clone program, or the COPY or WCOPY commands, to make a working cartridge). The clone routine is simply a series of COPY instructions, plus a Procedure which doesn't appear to be used. The program will run on a basic QL; for the review, a 640 KB QL was used, with free memory reduced to about 96 KB by formatting ram discs to take away the remainder. Loading and running took the usual several seconds from microdrive — nothing to frustrate the microdrive user.

You can play against either the computer or a fellow QL user. Alternatively, you can have the computer play itself, which is useful to get the idea of how to play. Illegal moves are rejected, and a low beep informs you of them. The board occupies about two-thirds of the left half of the screen; the top one-third of the right half contains the options menu, the lower two-thirds displays the moves, Red (bottom of the board) on the left and Green (top) on the right. 10 moves of each player can be displayed, and then the display is scrolled up 1. The menu offers the options TwoPlay, Hint, NewGame, Red, Green, Forward, Back, Other, Level, and Setup.

The "TwoPlay" opion per-

mits two users to play, and starts the game. The "Play" option starts the game, you-against-the-QL. You can let the computer move first, or make the first move yourself. (If the computer is to make the first move, it is said to be possible to make it think-out the move, rather than making the standard opening, but I didn't succeed in persuading the computer to do this.) "Hint" is just that — a suggestion, such as a possible move to make. "NewGame" allows you to start a new game, with all pieces being in their normal positions (as opposed to with the "Setup" option). "Red" causes the computer to move a piece on the Red side (bottom) when you choose to have the computer do all the playing; Red moves are made at Level 1 (see below). "Green" causes the computer to move a piece on the other side (top); moves are at the set level — ie the computer can be given more time to make Green moves than red ones.



When used after "Back", "Forward" makes the next move previously made. "Back" allows you to go back through the game, move by move, right to the start (if "Setup" has been used, you can only go back to the point the new setup was completed). "Other" displays the alternative board, which has different icons for the pieces. All the icons are well created, but the alternative set are particularly good, giving realistic representations of horse and rider, chariot, elephant, etc. at a small size where I didn't realise it was possible to have anything more than crude blobs. "Level" sets the computer's own response time; it can take up to seven seconds, 30 seconds, or one minute (Levels 1/2/3) to respond to your moves. The Level can be changed during the game. "Setup" allows you to set the board up in any (legal) configuration. By pressing ‹Space› and ‹E›, you can clear the board, ready to position the pieces where you want them; there is no option to save a game part-way through, so this option is needed. Menu selections are made by pressing the key for the first letter of the options.

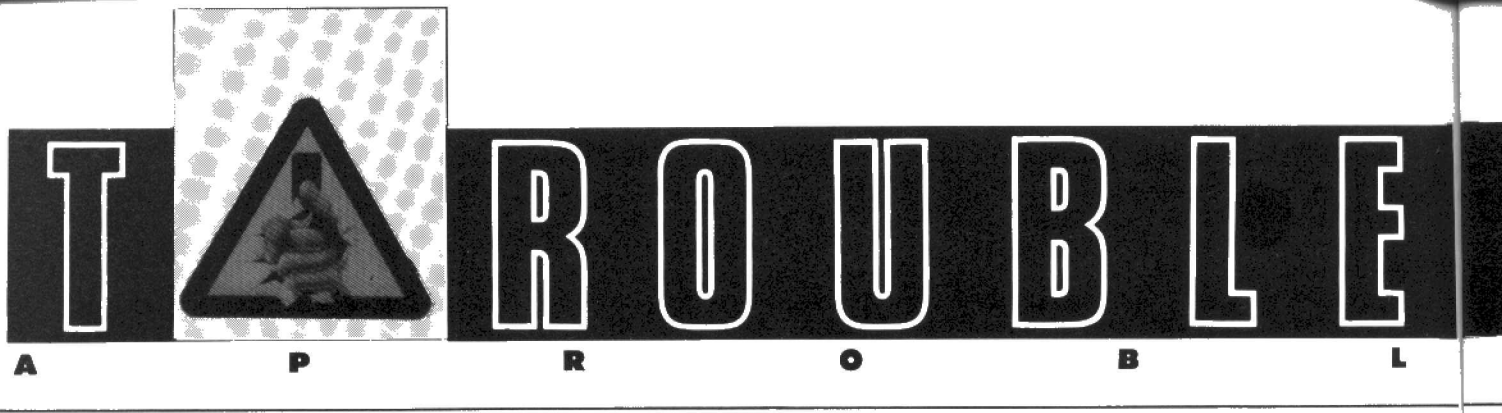
The moves of the pieces are signalled by the pieces flashing for 2-3 seconds,before and after the moves. This is a useful touch, particularly when learning the game by watching the computer play itself. Likewise, the ability to select playing Level allows you to "ease into" the game. A rather odd feature is that, when the computer plays itself, the moves do not have to be in the Red/Green sequence. In fact, you can go on pressing only ‹R›, or ‹G›, apparently until the game ends! Up to 10 moves, you can see if you have got out of synch in your keypresses, but the scrolling of the display then means you have to trust your memory (perhaps not a good thing, bearing in mind the typical age of us QL users?) Fortunately for the unmoving opponent, when the moves were made only with the pieces of the one player, there came a point where the same move was made repeatedly, and the game effectively came to an unresolved end.

After a move has been initiated, the menu options are replaced by a message, indicating what is going on; for example, "Red thinking". A beep from the speaker advises you that you have made a move (for those dozy periods when you are not quite sure what you are doing). Seven seconds may not sound a long time, but it begins to seem too long. There are times when it is obvious what the computer's next move has to be (sensibly), but it still takes the seven seconds to make it.

Icons

Overall, the screen presentation is good with the icons for the alternative (European-style) pieces being particularly good. Playing the game is not difficult, but neither is it so simple that it loses your attention. There is an irritating problem with response to the cursor and ENTER keys when moving pieces; you either have to hold the keys down longer than normal, or risk having to hit them 2-3 times. This may vary with QDOS version (a JM was used). On one occasion, the marker that indicates which player should move next, appeared on the wrong side of the board, and a "checkmate" situation was not indicated at the end of the same game. Neither "check" nor "checkmate" situations were indicated on other occasions. The first of the sample games had an incorrect move, there being no piece at an indicated location; when running through the other sample game, a lockup situation appeared to occur twice, but it turned out that the computer had decided to take control of the green pieces despite the game being started as a two-player one. The computer proceeded to play the next move-but one of the sample game! There was also an error in the stated moves for this game.

It takes quite a time to adjust to the significance of the Chinese symbols, as regards what their movements can be, but you can choose the more-descriptive (to European eyes) alternative set of icons instead. There were no problems with displays, and they were all clear and informative. There are some inconsistencies in documentation — for example, "Counsellor" appears on-screen as "Mandarin", and the description of the movements of the "Chariot" refers to it as a "Canon" — but they are not such as to cause the user great trouble. Overall, the drawbacks are not serious and should, hopefully, be ironed-out in future revisions. Apart from them, the game is interesting, clearly presented, works well, and is reasonably priced.



Bryan Davies considers disks and spell-checkers.

Two common features of letters we receive are spelling mistakes and the lack of QL-printed output. Without fear of being proved wrong, I'd risk saying that at least half the letters I see have spelling mistakes (that doesn't mean typographical errors). Much the same is now true of "printed matter" in general. The instructions for some QL programs are far from being models of either good spelling or clear thinking. Did I tell you that one about writing to the editor of *The Times* to complain about half a dozen spelling mistakes in one, short article on cars (it was a poor article anyway)? Back came a letter of apology, and an assurance that the problem was being dealt with. My name was spelt incorrectly on the letter!

How often do you see "seperate"? More subtle perhaps, but just as surely a mistake is the use of "complement" when "compliment" is meant, and vice-versa. This isn't intended to be a lesson in English, but a recommendation to consider buying a spelling-checker. The revised, verson 2, *SpellBound* is due out about now, and *QTyp* has been out for a long time (and is built-into *text* [87]).

There is, of course, one big difference between "separate" and "complement/compliment", when you come to spell-check them with a computer. You are (hopefully) certain to be told the first is a mistake, but both of the second words are correctly spelt; the error here is one of context — the way they are used — and few computer programs are yet capable of checking context. I'm using a 'beta test' copy of *SpellBound* 2 at the moment, and it seems to have overcome most of the objections raised about version 1. It is less likely to "drop out" of concurrent checking (and easy to get back in), a bigger dictionary is provided (50,000 words), saving the dictionary no longer means having to read it back in to carry on checking, and — above all — you can check an existing document. Retrospective checking is much faster than using the old comination of *SpellBound1* and *FileBound*. The "feel" of the program is similar to what it was before, so there is no great amount of re-learning involved in using it.

The QL-printing thing is interesting. Over the years, I think most letters which have not been complaints about suppliers have been on the subject of printing, yet many of our correspondents hand-write or typewrite their letters. My prime reason for buying a computer system was to get away from using a typewriter, but many others obviously have different reasons. Or is it simply that there are still many users who have printers but cannot get them to print properly? Repetition is something of a sin, we realise, but there are some computing tips which seem to need repeating over and over again. Don't be shy to ask for help if you have problems printing from your QL. Please do not expect anyone at *QL World* to write printer-drivers for you, though. We only have time to give general advice on how to go about the job.

In case readers interest in QL-to-PC file transfer read the review of *MS-QLink* in the June issue and wondered where the comments on an updated version of that program had got to, they are in the August *QL Scene*. All the problem areas mentioned in the review were dealt with by the version 1.3 update, and the latter was delivered to me in April. The program is now fully-functional, and is well worth the money. On quite a different tack alpha test versions of completely new program are being checked by the usual "destruction brigade"; the program looks promising, and should be of interest to most users. As to what it does, a recent editorial comment gave a large clue.

Italian

The Italian version of text[87] is now finished, and available. More ideas are being considered at *Software*[87], and we can expect further product announcements (but no dates have been put forward at this time). The author of *Home Budget*, which was reviewed in the May issue, has pointed out that version 2.0 was produced in February this year, and has some useful improvements in it. The sluggish highlighting of the menu choices has been cured, the "crash warning" when trying to load a non-existent ramdisk file has been eliminated, a bug was fixed, and the instructions were improved. Three menu options were added to the J-Tax part of the program; the Report facilities were increaesd, and the Capital Gains Tax file of RPI values was updated.

This example might serve as a warning to people who submit programs for review, to ensure that any updates are sent in promptly. For various reasons, there can be delays in reviews being published; make sure you are not sitting back admiring your latest, crackerjack version when the reviewer is cursing your first version. In this case, the author was unaware that his program had been sent in for review, as the matter was in the hands of the supplier that was selling it (well-featured in this month's complaints!). So, keep checking the supplier, if you put your program in someone else's hands. Better still, don't put your program on the market with bugs in it.

Notwithstanding previous comments about taking care when buying cheap disks, I decided that 38 pence each for 3½in disks in lots of 25 was too good a price to ignore, at the All Formats Computer Fair on June 9th. Disks said to be of Sony manufacture, but "unbranded", were on sale at 40 pence in lots of 10, and there is a £2.50 post and packing charge (but VAT is included). The normal price for 5¼in DS/DD is 22p each, plus post and packing, in 10s. It is always some time before you know whether or not disks are good ones, and I can't make any comment on the quality of these, other than to say they all formatted without problem to 1.44 MB on the PC/AT, and will obviously format to 720 KB on the QL. The are currently being used to backup a hard disk and are giving no trouble. The main reasons I though it worth trying them was because the supplier selling them was prepared to take credit cards for payment was giving receipts without prompting, and supplied both a price list and visiting card with full name and address on (see INFORMATION). Their price list states "No quibble replacement or money-back guarantee". That indicates some degree of trustworthiness, although I am very conscious of the way some suppliers in the past looked good at first, then turned distinctly unreliable.

The usual stalwarts were at the June Show — Miracle Systems, Digital Precision, Sector Software, Tony Firshman, EEC, Quanta, and Liberation Softward were joined by QView. The activity level at the QL stands looked reasonable, although it seems that the stand prices are now appreciably higher than they used to be and quite a bit of "gear" needs to be sold to break even. There were some grumbles afterwards that the numbers attending were down.

SHOOTER

EMSOLVED

Readers Letters

Some comment has been supplied on the CST SCSI (hard disk) interface. The instructions were basically the same as those supplied with the floppy disk interface, but there was an extra page with details of the revised Format command. The toolkit commands in the interface were "never fully documented", but you might be able to identify what they are by using the <Extras> command to list the available SuperBasic commands.

Has anyone got a copy they could spare of the instructions for a Silicon Express interface and disk drive set? F. M. Johnson has these units, and is unfamiliar with using them; unfortunately, the instructions were apparently supplied as a disk file, and that has become corrupted.

The complaints about PDQL continue to come in; bear in mind that there is a delay of several months between letters being written to *QL World* and my comments on them being printed here. As of mid-June, PDQL had called and advised that all outstanding matters were being dealt with, but it has since been reported that even close associates have been unable to get in touch with the company. Robert Freeman reports not receiving *Turbo-Plus Quill* from PDQL, and not getting his money refunded when he finally cancelled the order. There appears to be doubt about the ability, or willingness, of the originator of Turbo-Plus to supply copies, however; another reader, G. Perrett, complains that T.K. failed to supply this program, but I believe they accepted this and other orders in good faith, believing they could obtain the program, only to find they could not do so in a reasonable time.

Another reader who order Turbo-Plus from PDQL and got nothing is Tony Morgan. G. M. Young ordered the same program, and has also been waiting for it, or a refund, for some months. Yet another would be purchaser of Turbo-Plus is W. Mason, who has now had to contact his credit card company, having failed to get any satisfaction from PDQL; he also ordered (and did not receive) *SpellBound* from them. Donald Armstrong failed to obtain the ordered software from PDQL, but did have his money refunded by TSB TrustCard, and has since had 24-hour delivery after ordering the same software (by fax) from T.K. Computerware. Scott Telford place an order for the PDQ-C compiler last October, and his cheque was cashed promptly, but he has since obtained neither the software nor any satisfaction after "numerous phone calls" to PDQL. Michael Squires ordered *Multi-Discover* as recently as March, and even got a promise of delivery by hand, but has received nothing.

Serious

Two complaints which sound much more serious came form M. Burden and R. Dand. Both of these *QL World* readers report having had charges made twice to their credit card account for the same item. Burden sent his Thor for repair by PDQL in July 1989 and had not got it back as of 20th June this year; his account has been debited with about £340, but the repair charge was only £148. Dand actually received the ordered goods (a dual disk drive unit but his account was debited for £360 instead of the quoted £180. Both these readers have been advised to make claims against their credit card companies, as soon as possible. They stand a much better chance of getting their money back than do those who have paid by cheque.

A. E. Le Feuvre sent a copy of a letter to T.K. Computerware asking for a large refund on software, but didn't advise us as to what the software or the problem was. T.K. stated that the customer had called to say *PC Conqueror* "does not work", almost immediately upon receipt of the program, and insisted on return of his money. As this program does not have any operation problems so far as we are aware, the reason for return is a mystery. In any event, T.K. had refunded the money some time before being contacted by us.

The complaints of J R were referred to last month, and T.K. commented that they have twice sent replacements of *TechniQL* to him, and it would appear he has received at least one copy, judging by his comments on the later version of the program, made to another magazine. It seems likely that he either has a problem with his printer, or it is not fully "Epson-compatible".

On this later point, that phrase has a restricted meaning in most instances; it refers only to compatibility with the Epson FX-80 printer, and even several of Epson's own printers are far from compatible with that. Goodall appears to have had or requested, more than one replacement printer interface also, and this reinforces the thought that the problem had not been traced to its source.

Transform report sending Goodall a cheque as refund for the *Organiser dBase* link set to him in error, and they have supplied him with a *Datapak* which was outstanding from an earlier order. They have also replaced his Organiser Centronics interface. Goodall confirms having received the items sent by Transform, but still reports a problem using TehniQL. He cannot get on the screen the prompt which asks which printer port to direct output to. Miracle Systems have replied about another of his complaints, saying that they have been out of stock of serial-parallel interfaces fitted with the 9-pin "D" connector and have therefore been unable to deal with his order for one. Unfortunately, they failed to write to him and explain this. A voltage regulator ordered by D. Thomas in January was not sent out by T.K. until mid-March, due to factors beyond their control.

In response to David McKail's request for information, Graham Priestley (ex-CST) says he also uses *XChange*, it looks as though they never really upgraded it at all, and McKail should look elsewhere for the source of the problem. David McCullagh (a correspondent of long standing!) is still having problems with his Thor XVI and says that Thor International (Dansoft) "are uncontactable and unhelpful". It is over a month since I wrote to this supplier asking for comment on the general situation with the Thor XVI and no reply has been received yet. McCullagh states that the Argos rom in his machine has a bug which make *The Editor* unusable, and he is no longer able to use the Mircale serial-parallel interface with his printer, apparently because the XVI has a (PC-type) parallel outlet.

INFORMATION
MS-QLink v1.3, price £12;
Qfile
Apartado 2110
1103 Lisboa Codex
Portugal

Cheap disks:
MediaVALUE
Northumberland House
Drake Avenue
Staines
Middlesex TW18 2AP
Tel. (0784)-466744

SUPER BASIC

The Minerva ROM has been quietly taking root in the QL community. Mike Lloyd finds out what it's about.

For QL users who do not subscribe to Quanta the first to be heard of the Minerva rom was Simon Goodwin's article in the November 1989 issue of *Sinclair QL World*. Since then the product has undergone development to reach its present state. What does it have to offer SuperBasic programmers?

The Minerva rom comprises a tiny circuit board which fits in place of two of the normal QL rom chips. Fitting requires the top to be removed from the computer but the process is simple and painless provided it is undertaken with care and patience. Once in place it works in exactly the same way as the roms it replaced, providing the intelligence which gives the Motorola 68008 CPU the characteristics of a QL rather than, say, those of an Amiga or an Atari 520.

The main difficulty facing the QView team responsible for Minerva was how to make the new rom better than the originals while ensuring that the revisions remained compatible with existing standards. Without this compatibility many pieces of software would not run with the Minerva, and its value would be curtailed. Almost by definition, no improvement can be fully compatible with what it is designed to replace, and this is true of Minerva. However, fixes exist for the few problem programs — some provided to QView and others by the programmers of the offending pieces of software.

QL owners who use their machines exclusively with commercial software know already that Minerva enhances their machine's speed and reliability with only minor unwelcome side effects. But what of the thousands of QL owners programming their own computers: how are they affected by the arrival of Minerva?

Program execution speeds are accelerated due to faster graphics routines, the implementation of integer loops and the halving of the time taken to jump to a procedure or function definition in large programs. Arcade-style games written in Basic can, with little change to the program design, be made to run at up to twice the speed of the original. However, the greatest benefits come when writing programs specifically for use under Minerva.

When ERRor

A recent Super Basic article examined in detail the WHEN ERRor code implemented by both Minerva and *Toolkit 2* which made file handling in particular much more controllable. The WHEN ERRor construct allows programmers to write routines to ensure that any run-time error can be corrected without crashing the program. I have been asked to point out that, because Minerva rewrites the code other programs cannot reach, as it were, its WHEN ERRor routines are likely to be more robust than Toolkit 2's, especially outside the area of file handling. This is not because of any weakness in TK2, but simply that TK2 provides extensions for Qdos whereas Minerva rewrites Qdos itself.

Several quite minor changes made by Minerva to SuperBasic's normal behaviour are undocumented by QView because, they say, the effect is slight or because the changes actually bring SuperBasic more into line with the *User Guide*. Some of these changes, unfortunately, are immediately counteracted by *Super Toolkit 2* which, under the impression that it is eradicating poor routines from the original roms, actually by passes the perfectly reliable and even enhanced code in Minerva. Readers might care to write in with their finds.

Some features of SuperBasic are carried over into Minerva unchanged. For example, the statement:

```
PRINT TO 0; "HELLO"
```

Actually prints the string indented one character from the beginning of the line. This might be annoying, but it is consistent with the rule that when asked to print at or to the left of the current print position the QL will begin printing one character place beyond the current print position. It can be awkward, though, when printing must begin at the start of a line under the control of a variable, such as:

```
FOR POS = 0, 10, 20:
PRINT TO POS; "X";
```

Under both standard Super Basic and Minerva the above must be rewritten:

```
100 FOR POS = 0, 10, 20
110 IF POS = 0
120 PRINT "X"
130 ELSE
140 PRINT TO POS; "X";
150 END IF
160 END FOR POS
```

Other features retained unaltered are the KEYROW bug reported in the User Guide which can deceive the computer into thinking that four keys instead of three have been pressed. The deadly CTRL-ALT-7 combination still locks up the QL and, incidentally, prevents the soft reset from the keyboard which Minerva implements. The WIDTH command, which did nothing on the old QLs, still does nothing under Minerva.

Cursor

On the other hand, Minerva puts right the incorrect behaviour of the CURSOR command. Originally, CURSOR could confuse the interpreter if it was followed by a channel parameter. Directed to the default window, CURSOR worked exactly as explained in the User Guide. If the command was followed by two parameters it used the pixel co-ordinate system to move the printing cursor. With four parameters the first two described a point in the graphics co-ordinate system and the second pair were offset from this point measured on the pixel system.

However, if the first parameter was a channel number and there were a further four parameters, the interpreter objected to what it believed to be an incorrect number of parameters. The workarounds were to do without the last co-ordinate or do without a channel reference. Minerva corrects this irritating bug and CURSOR now behaves itself in all windows.

The graphics routines have been completely rewritten for

Minerva, with the result that they are considerably faster than they used to be, and indeed almost rival *Lightning* graphics for speed. The old QL took 75 seconds to draw 1000 diagonals across the screen, but with Minerva the same procedure takes 50 seconds without taking advantage of an integer FOR . . . NEXT loop.

Interestingly, the acceleration has not been uniform. In an early *Better Basic* article it was reported that the BLOCK command was about three times faster at drawing horizontal lines than was the LINE command. Both BLOCK and LINE have been speeded up by Minerva, but BLOCK now outpaces LINE by a factor of five.

The relative speeds of the IF and SELECT commands have also been affected. Under some circumstances, SELECT used to be slower than IF — where, for example, there was a single expression and it proved to be untrue — but now SELECT seems to be faster than IF in all circumstances. The most significant difference is for range expressions which are untrue. If T equalled 100 and were tested by the following expressions:

```
100 IF T>= 50 AND T =<
70: PRINT T

200 SELECT ON T = 50 TO
70: PRINT T
```

the SELECT variant would be about 40% faster, although in most programs both commands are so fast that the difference hardly matters. Games designers faced with a profusion of multiple choices and anxious to squeeze every last percent of performance might well replace IFs by SELECTs, though.

A bug in the old FILL command could be exploited to produce interesting patterns using a simple routine such as:

```
100 FILL 1
110 REPeat loop
120 CIRCLE RND(100),
RND(100), RND(50)
130 END REPeat loop
```

With Minerva fitted, FILL has a different, but no less unusual, effect which I leave readers to discover for themselves.

Sadly, one of the means by which Minerva achieves an improvement in the QL's performance also leads to a slight problem. Minerva tokenises all integers to reduce program space and to improve performance, both by a factor of between 10% and 15%. However, as a direct result, the RENUM command no longer corrects line number references in RESTORE, GOTO and GOSUB commands if the line they refer to is numbered less than 127. This, says the Minerva manual, should rarely be a problem — unless, that is, you assume that RENUM works as before and have a RESTORE pointing to Line 120 and cannot understand why the program falls over after being renumbered. The Minerva guide includes details of POKE which will disable integer tokenisation and thus restore predictable behaviour.

Another potential difficulty is the way that the Super Toolkit 2 commands WTV and WMON work. The TK2 manual states that they should take a parameter similar to the MODE command, "4" for high-resolution screens and "8" for low-resolution screens, but most users quickly find out that the commands also work without any parameters. Under Minerva, though, the parameters become essential if the commands are to do what was intended of them.

Sufferers

Some QL users have reported that with Minerva fitted they experience problems with the printer buffer similar to those encountered when an incorrect baud rate is selected. Characters are missed out or misinterpreted by the printer as control codes, thus destroying any document. The corruption is intermittent in that a reset usually restores accurate communications to the printer, at least until the next time that the QL is powered up.

The people suffering most with this problem are likely to be those with Miracle serial-to-parallel connectors, especially early versions, and those with the printer driver from Qram installed. QView report that standard QL baud rates can be substantially lower than the BAUD command parameters would suggest, so that the QL transmits at about 8000 baud instead of the default 9600. Some software and hardware are geared to this reduced performance and have been rudely awakened by the sprightly Minerva, which just about reaches the correct transmission rate.

Solutions

Possible solutions for Qram owners is either to remove the "ramprt" drivers from their system or to direct the spooler to a dummy device rather than to "ser1", but neither option seems to be a watertight cure. No doubt someone will find a clever workaround and publish it in QL World.

If these are the drawbacks to fitting Minerva they are insignificant compared with the advantages the new rom has to offer. Minerva is faster, more reliable and has more features than standard QL roms, making it a desirable upgrade for programmers and non-programmers alike. The list of facilities and improvements of particular value to SuperBasic programmers is mouthwatering.

The changes to the line editing regime might not be as dramatic as those wrought by the "ED" keyword in TK2 but they are none the less welcome. The cursor can be dragged to the beginning or end of a line by combining the left or right cursor key with the ALT key. Alternatively, the cursor can hop through a long line eight characters at a time by pressing TAB or SHIFT-TAB.

In line with standard QL practice, adding CTRL to an ALT-cursor combination deletes from the current cursor position either to the end of the line or the beginning of the line depending on which arrow key is pressed. The ESCape key now performs the break function normally undertaken by the CTRL-SPACE combination and the annoying refusal of the QL to accept SHIFT-SPACE is cured once and for all.

Interestingly these improvements are also available to the INPUT command because AUTO, EDIT and INPUT share the same character-entry code.

Minerva is much more relaxed about slicing text and numbers than was the original QL, with the effect that just about anything, including the string returned by DATE(), can be sliced up and delivered piecemeal to a variable or to the screen. The command:

```
PRINT DATE$() (1 TO 4)
```

now prints the year rather than an error message.

Sinclair's idiosyncratic handling of strings received much criticism from people more used to Basic dialects more in line with Microsoft's standards, but their criticism was laced with envy because Sinclair string-handling is incomparably neater and more efficient. However, that does not imply that it was beyond improvement. For instance,

```
PRINT X$ (TO 6)
```

normally fails because of the lack of a starting parameter: with Minerva this weakness is removed. Similarly, programmers often need to pare strings down character by character until all that is left is a null string. This process is greatly simplified using Minerva because where B$ is a null string PRINT B$(1) no longer provokes an error.

The code governing the SELECT and FOR . . . NEXT structures has been changed so that it copes with integers and, to a limited extent, with character strings. The advantage of integer loops and SELECTs is one of speed, while character strings can add to elegance and simplicity.

The following program fragment demonstrates the advantages of using a string parameter in a SELect structure:

```
100 REPeat loop
110 d$ = INKEY$()
120 SELect ON d$
130 = "a": PRINT "A
pressed"
140 = "b": PRINT "B
pressed"
150 = REMAINDER: PRINT
"---"
160 END SELect
170 END REPeat loop
```

There is no longer any need to convert D$ to its ASCII code, nor is there a need to cope with upper and lower case variants — ON d$ = "a" matches both upper and lower case "A"s equally well. The result is faster, more elegant and easier to read.

Many programs make intelligent use of the system variables to enhance their abilities, but under QDOS the system variables are likely to be shunted around the available RAM as it fills up. On unenhanced QLs the system variables remain in one place, but it is unwise to depend upon them staying there if extra memory or additional devices are fitted. Minerva implements a revised VER$ command so that LET BAS = VER$(−1) reveals the start address of the system variables area. Even better, the PEEK and POKE commands have been improved to make it easier to navigate through the maze of pointers and offsets which characterises QDOS.

PRINT PEEK(!!55)

prints the value found 55 bytes from the beginning of the system variables area, while

POKE !124!51, 76

seeks out the byte offset 51 places from the start of the table pointed to by the vector located 124 bytes from the beginning of the system variables area. Uses for this powerful facility will be demonstrated in full in a future Super Basic.

The word and long-word variants of PEEK and POKE (eg PEEK_W AND POLE_L) can now access odd-numbered addresses without generating an error. This is of no use in the System Variables area where all long addresses are correctly located at even-numbered addresses, but it might well be useful when directly accessing the display map. POKE can now be followed by a long list of values which will be placed in the bytes following the declared start address, so that the single command:

POKE add, 20,30,40,50

replaces four old POKE commands.

The crowning glory of the most recent versions of Minerva is the ability to multitask many Basic programs. This has been a long-promised feature, so it is almost an anti-climax to find that the process is really extremely simple. QView dispatch with each Minerva a disk or microdrive of supplementary programs which include one called "multib_exe". Every time this short file is executed a new SuperBasic job starts. The code can be made resident and linked to a hotkey if the programmer wishes. Movement between SuperBasic jobs is via the usual CTRL-C combination.

Windows

When it is first opened, a new SuperBasic task has but one window of limited size performing the duties of both command and default window. It is up to programmers to open other windows and resize them to emulate the normal SuperBasic screen layout if they wish. By default all extensions to SuperBasic owned by the calling job (normally standard SuperBasic, but possibly a second SuperBasic process) are inherited, but programmers can opt to start a SuperBasic with just the standard set of keywords. All Basic jobs are interrupted by pressing CTRL-ALT-SPACE rather than CTRL-SPACE or ESCape. Communication between tasks can be achieved using the EX keyword (from TK2) with parameters. Data can also be piped between tasks using the PIPE facility from Turbo Toolkit.

Minerva also offers other enhancements such as true TRON and TROFF code-tracing commands which are invaluable when debugging recalcitrant software and new syntax options for ABS, ATAN, MODE, PAUSE, SCALE and DATE. All in all, it amounts to an irresistable package at a very reasonable price. QL culture is not a dead world: with Minerva the QL takes another leap forward in its extraordinary history.

SuperBasic and Minerva

Since Super Basic's predecessor, Better Basic, first appeared more years ago than I care to remember all of the routines and complete programs that have been printed have been designed meticulously to ensure that readers with unexpanded QLs linked to TV sets were not disadvantaged compared with their monitor-viewing, memory-expanding, Turbo-compiling, disk-saving, high-spending neighbours.

Pressure

The pressure to modify this policy is now too strong to ignore. Most QL programmers have access to a toolkit of some sort, be it Tony Tebby's Super Toolkit 2, Digital Precision's Turbo Toolkit, or one of the others available both in the UK and abroad. Something like 60% of QL users are believed to have memory expansion fitted to their machines and the number with disk drives supplementing microdrives is almost equally high. The most recent development is the arrival of QView's Minerva rom which replaces the Sinclair rom, enhancing many SuperBasic keywords and providing some powerful additional features.

When Super Basic was launched in the September 1987 edition of Sinclair QL World the majority of QL owners were content with the standard 128K machines and it was therefore important that Super Basic's programs worked within the limitations of the majority's machines. However, as the QL market has inevitably shrunk the stayers have been those who have spent money on their QLs, so now the majority of users do not *have unenhanced machines. It seems inconceivable that there are more than a handful of keen SuperBasic programmers who do not have Super Toolkit, and so it now seems incongruous for Super Basic to pretend it does not exist.*

The same is going to be increasingly true for the amazing Minerva rom replacement which has quietly been occupying chip sockets vacated by bug-ridden, lethargic and unfriendly Sinclair roms in QLs throughout the country. The changes wrought by the Minerva rom are much too important to ignore, particularly when they add so many features for which SuperBasic programmers have been wishing, such as true integer arithmetic, integer loops and string SELects. Almost as an added extra, recent Minervas permit many SuperBasics to exist simultaneously

Minimalist

The unrepentently minimalist SuperBasic programer need not fear that all subsequent Super Basic articles will not work on his or her machine: unless there are strong reasons for using Minerva or Super Toolkit 2 features the programs will continue to be usable on unadorned QLs. Where possible, too, standard SuperBasic solutions to problems will be listed next to the Minerva or TK2 solution. Inevitably, however, Super Basic routines taking advantage of new features will be neater, faster, more robust or more concise than the standard SuperBasic equivalent.

The QL was never meant to remain in one set specification: the operating system and the SuperBasic language were deliberately designed from the start to be flexible and extendable. Although "official" development of the QL stopped with the Amstrad takeover of Sinclair's rights the QL has never stopped growing and improving.

Contender

Indeed, the QL has already solved all of the problems currently besetting MS-Dos computer designs and, in a fair world and judged purely on merit, it should have been a major contender in the PC marketplace. There may one day be a better personal computer than the modern QL, but I doubt whether there will ever be one to match its value for money. If computers ever come to share the same status as old cars you can safely bet that the Sinclair QL will be one of the classics.

Programming IN C

Andy Wright begins a series of articles on programming in the machine-flexible, high-level language C. Even beginners can benefit.

This article and the ones which follow it will give you enough information to be able to start writing C programs.

It is assumed that you are already familiar with some other programming language (SuperBasic, for instance) and that you are not a complete novice — but you needn't be an expert! It is also assumed that you are reasonably familiar with the use of a QL.

For a complete description of the C programming language you could do worse than get hold of a copy of *The C Programming Language* by Brian W. Kernighan and Dennis M. Ritchie (Prentice Hall ISBN 0-13-110163-3).

The examples given have all been fully tested using Digital Precision's *Digital C* compiler. The coding should also work with other C compilers unless otherwise noted in the text.

C started life as a programming language used on computers running the Unix operating system. Today it is available on almost all personal computers, mini-computers and mainframes — probably covering as wide a range as any other programming language.

Whether or not C is seen to be a good programming language seems to depend on who you ask. Some of the pros and cons are summarised below:

For:

1) C is a (fairly) high-level language, so is easier to learn and write than assembler. However, many facilities are provided in C that are normally only found in low-level assembly languages. This makes C unusually versatile.

2) C generally produces fast and efficient programs.

3) C is a compact language — you can do a lot with very few lines of code. This can make it attractive to programmers.

4) C is much the same on many different computers. This means that programs can often be transferred from one machine to another with a minimum of fuss and effort. For example I have coded programs on the QL, transferred and used them in an MS/DOS (IBM PC compatible) environment (the operating system used on ICL mainframe computers) without problem. I cannot think of another programming language where this would have been so straightforward.

Against:

1) C programs are often difficult to understand if you don't know C — the coding is often far less obvious to the uninitiated than is, say, SuperBasic or Cobol or many other languages

Full and complete versions of C are based on the definition produced by Brian Kernighan and Dennis Ritchie.

Many versions of C are based on a subset of the facilities described by Kernighan and Ritchie — these are generally known as Small C. Some of the more powerful (and perhaps) less used facilities are not included in small C *Digital C* is a small C compiler. However, some facilities associated with the full Kernighan and Ritchie implementation are provided as extras.

This series of articles is based largley on Digital C. The extra facilities provided by Digital C are also explained.

Another C compiler available for the QL is Lattice C which is advertised as a full implementation of the language. A brief discussion of the extras (which go beyond small C) is included later in the series.

In order to write a C program you need a text editor of some kind. For example, *The Editor (Digital Precision), Spy (ARK), or Micro Emacs (Quanta).*

The text editor is used to create a file containing the C program code. The (source) code in this file is then compiled to produce a machine code program. The machine code program can then be run as required.

All of the examples in these articles are of C source coding. This needs to be stored in a file using an editor. The C compiler is used to convert the source code statements into machine code.

How to go about compiling the program depends on the compiler that you are using. The example in this section uses the Digital C compiler.

A C program consists of a collection of one or more function definitions. A function is a distinct piece of code (rather like a SuperBasic function or procedure).

The main part of a C program consists of a function called Main. This has to be the first function that is defined in the program. An example of the basic format of a C program is given in **Fig.1**. Don't try and compile the coding in Fig 1!

```
main ()     /* a comment */
{
 c_statement;
                /* another
                   comment
                */
 c_statement;
 c_statement;
}
```

Fig. 1 Basic structure of a C program.

main () defines the main program. As mentioned above, C programs need a main function. The empty round brackets () indicate that the program does not expect any parameter (argument) values to be passed to it when it runs. The brackets have to be included! Note that the word main must be entered in lower case.

The coding for the main function is always enclosed between curly brackets {and}.

C statements are included between the { }. Each statement must be terminated by a semi-colon (;). You can separate statements with blank (empty) lines to improve readability.

The lines of coding can also be indented to make it easier to read.

Comments can be included — they begin with /* and end with*/. C compilers ignore everything that is enclosed within the comment delimiters /* and */.

Fig. 2 is an example of a very simple C program.

The program executes two statements. Each statement is a call to the printf function — this is a standard C function. The standard C functions are provided with the compiler.

Parameters to functions are always enclosed within round brackets. In **Fig 2** the printf function is passed a single parameter — a string of characters enclosed in quotation marks. Note that double quotes (") and not single quotes (') must be used. Printf displays these characters on the standard program output device (the screen). The word printf must be typed in lower case.

Note that the first call to printf includes the characters n in the string that is displayed. Characters preceeded by a are known as escape sequences and have a special meaning. Some of them are:

n/ print a newline character
t / print a TAB character

```
main ()
{
    printf ("my first C program ndisplays two lines");
    /*
        printf must be in lower case
        you can leave spaces between printf and the opening (
        you can leave spaces between the ( and the opening quotes
        you can leave spaces between the closing quotes and the closing )
        spaces between the quotes are printed as spaces
        you can leave spaces between the closing ) and the semi-colon the
          whole of this comment text can be omitted
        no newline character is printed at the end of the printed text unless
          you use n at the end

    */

    printf ("of text");

}
```

Fig. 2. A simple C program

f / print a FORMFEED character
print a character
b/ print a backspace character

The output from the program is displayed is displayed on the screen as:

My first C program
displays two lines of text

(a newline is printed between the words program and displays).

If the source program is edited into a file named flp2_test_c (Digital C insists that C sources programs are stored in files that end with _c) then it can be compiled as follows:

With the disk containing Digital C on device flp1_use:

exec_w flp1_cc (ENTER)

After a few seconds, when prompted for a command line enter:

flp2_test _p _m (ENTER)

This tells the first stage of the compilation process (the parser) that the source code is in a file name flps_test_c, that it should pause if there are any errors (it waits until you press ENTER), and it should monitor its progress — it displays messages as it does things. Note that the -c doesn't need to be specified at the end of the source file name.

This should produce an object file named flps_test_obj. The compiled machine code is then generated by using:

exec_w flp1/cg (ENTER)

when prompted for a command line enter:

flp2_test_exe flps_test -m -p -nc -d/flp1_(ENTER)

This tells the code generator to produce a compiled program in a file named flp2_test_exe. It reads from the object file names fl_2_test_obj (the _obj doesn't need to be specified.) It monitors its progress and pauses if there are errors. There will be no prompt for a command line when the final program is executed and the standard C libraries (provided with Digital C) are on the deivce flp1_ on the same disk as the Digital C compiler itself.

The diagram in **Fig. 3** illustrates the compilation process:

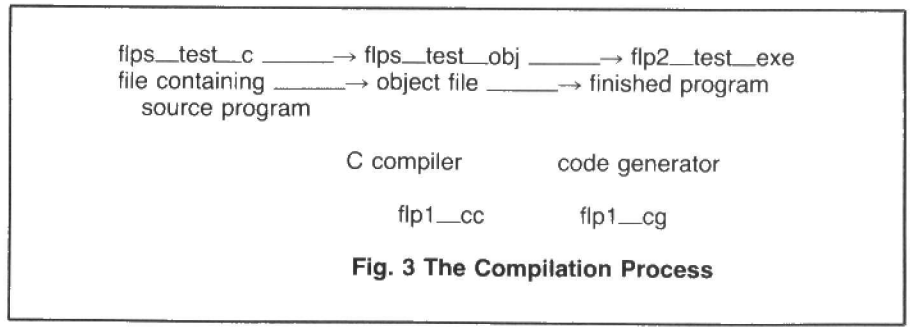


Fig. 3 The Compilation Process

To run the finished program use:

exex_w flp2_test_exe (ENTER)

Note that the exec command can be used instead of exec_w. The digital C manual also explains how the ex command (in SuperToolkit II) or the execute command (in the Turbo toolkit) can be used to run the compilers. It is, in fact, possible to build a fairly neat front end program to allow compilations etc to take place at a single keystroke. I use such a system myself since it saves a lot of time.

Also note that if you run the final program in, say, a Qram environment the output display is immediately cleared as soon as the program terminates — this tidying up of the display is handled automatically by the Qram window managment software. Future articles will illustrate how this can be overcome if it causes problems.

Some follow up work

Try some of the following:

1) Add some more printf statements to make the program display more messages. Also add a few comments.

2) Try using some of the other escape characters to see what effect they have.

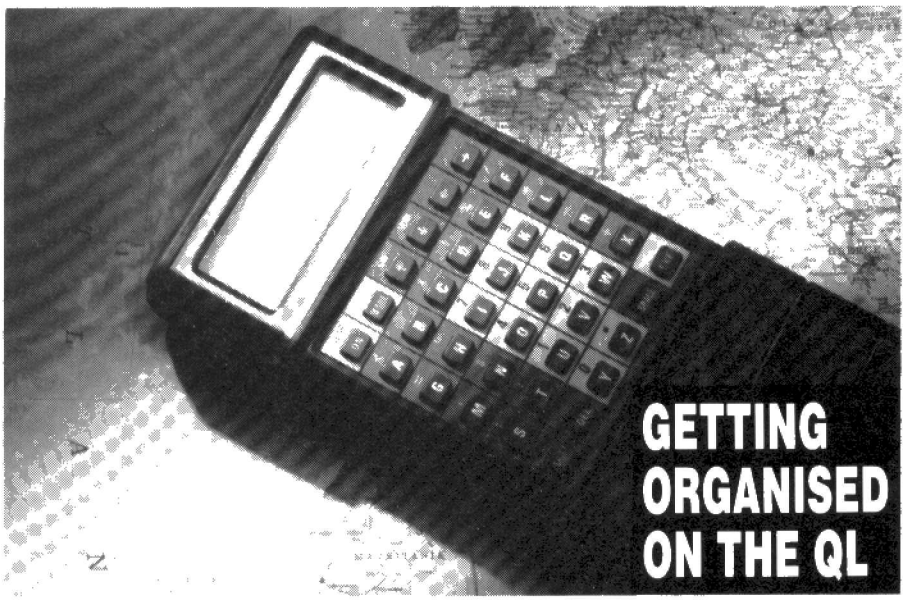
3) Try splitting the printf statement over two lines — it should work ok providing you do not split the quotation marks and the enclosed string of characters across lines.

4) Misspell the word printf to see what kind of error message(s) you get from the compiler.

5) Other errors could be generated by:

misspelling the word main
missing out one (or both) of the brackets after main
missing out one or both of the curly brackets
missing out one or both of the quotes in a printf statement
missing out one or more semi-colons from the end of statements
splitting the quotes and/or the string of the printf statement across two lines

Although it may seem stupid to deliberately add errors to programs — a little bit of time spent doing it now could save you ages and ages when you accidentally include errors some time in the future! *(Hear, hear — Ed.).*



GETTING ORGANISED ON THE QL

David Drysdale lists a few ways of getting facts on file for day to day use.

Pocket organisers, despite their Yuppie image, are of tremendous help to busy people. There are two types to choose from — the electronic ones, such as those made by Psion, or the 'Filofax' type with paper pages. The QL works well with both of them.

Put simply, the art of organisation is centred on making lists — usually of things to do — and the QL is ideal for producing such lists and presenting them in an organiser compatible form. Then the QL can either export them to an electronic pocket organiser or produce a paper printout for placing in a loose-leaf binder.

It is the loose-leaf type of personal organiser that is most widely used and it is worthwhile looking through the pages of some of them to see just where the QL can be of use.

An ordinary diary usually forms the basis of an organiser system and this often has an appointments section with timetables for the day. This can list activities such as phone calls to make, people to see, letters to write or things to do. Year planners are available too.

Because there is such a wide selection of organiser diaries, both from Filofax and their competitors, it is probably best to buy one if a paper-type organiser is being used. The more personal lists such as things to do, calls to make and letters to write could, of course, be produced on the QL and printed out but, unless one is using the QL every day, the old-fashioned pencil would be just as quick to use.

Information pages, usually indexed, form the next largest block of organiser leaves and it is here that the QL comes into its own. These pages, particularly of addressed and telephone numbers, require constant updating and *Archive's* power to delete, alter, select, order and print are most useful. Most QL users already have this kind of information stored in Archive and can save themselves the laborious chose of writing it out in longhand on blank paper sheets.

The most useful pages in my own loose leaf organiser are the telephone list. I did not have these Archive pages stored as a telephone directory because the phone numbers were listed on the pages of my computerised address book. However, a simple instruction to Archive to select names and telephone numbers from the address book and to print them out soon produced the list I needed. I then exported the list to *Quill* for easy page make up to fit the organiser.

This method of printout gives me 36 names and addresses on each page with the Quill printer drivers. Most lines could be inserted using a word processor such as *Text*^{87} with variable line spacing.

If it is aimed to pack in the maximum information on your organiser leaf the answer is to design a larger Quill page that can be reduced in size on a photocopier. Bigger sheets of paper can also be inserted into a personal organiser by having them fanfolded into two or three sections and this way quite sizable spreadsheets from Abacus can be photoreduced to fit allowing all kinds of tables, budgets, and other pages of financial information to be produced for the organiser.

The other, and perhaps largest, category of organiser leaf is the most specialised "printed form" designed for, say, keeping photographic records. Again it might be more economical to buy the ready printed version but one's own needs may suggest a different approach. A photojournalist, for example, may need his subject's telephone number recorded but not find a space for this information on any read-made product and custom made forms and records could be the answer.

If it is aimed to pack in the maximum information on your organiser leaf the answer is to design a larger Quill page that can be reduced in size on a photocopier. Bigger sheets of paper can also be inserted into a personal organiser by having them fanfolded into two or three sections and this way quite sizable spreadsheets from Abacus can be photoreduced to fit allowing all kinds of tables, budgets, and other pages of financial information to be produced for the organiser.

The other, and perhaps largest, category of organiser leaf is the more specialised "printed form" designed for, say, keeping photographic records. Again it might be more economical to buy the ready printed version but one's own needs may suggest a different approach. A photojournalist, for example, may need his subject's telephone number recorded but not find a space for this information on any ready-made product and custom made forms and records could be the answer.

The actual printing out of the organiser sheets can be done one at a time using the easily obtainable perforated blanks or it can be done on plain paper which is performated afterwards with a six-hole organiser page punch. There is a plastic punch on the market which sells for less than £3. Sprocketed continuous fanfold stationery packs with ready punched pages of the exact size are also sold under the trade name of Daatafax.

Single and double leaf continuous stationery packs can also be bought from Filofax, and a useful first step for anybody planning a personal organiser could be to buy a copy of their Filofax organiser catalogue.

Catalogue

This catalogue highlights a vast range of organiser leaves including their free-form Time Management system. This system has more than thirty different application leaves and a booklet containing some of them and explaining the philosophy behind the system can be purchased from Filofax suppliers

Another useful book for planners is the *Complete Book of Household Lists* by Nina Grunfeld which could probably be obtained from the public library. This book explains how almost everthing can be organised by making a simple list. Amongst the lists given are: organising a tool box, selling a car, how to save money, children's party games, spring cleaning, and — perhaps the most useful of all to some people — getting rid of a hangover!

One piece or organiser software I looked into was *Filerfacts* by SD Microsystems. This was not designed for printing organiser leaves or exporting to electronic units but for those intending to use the QL itself as an electronic desk diary.

The program was Archive-based but if the user asked for a spreadsheet-type

application the program would ask for the Abacus cartridge to be placed in the machine. The program included a desk diary, appointments list, budget planner, reference file and a main menu from which to select the applications.

The reference file was a simple Archive-based card index with one-key instructions to (B)rowse, (F)ind, (I)nsert etc. but nothing in the package is likely to impress any sophisticated QL user.

Iput this point to SD Microsystems who told me the program was not designed for high-level users but for the many newcomers to the QL who have great difficulty in understanding *Archive* and *Abacus*. The program has been designed as a useful applications pack which, at the same time, would help new users to find their way around the Archive and Abacus packages.

Transfer

At the moment people wishing to transfer files from the QL directly to an electronic pockets organiser have only two units to choose from: The Psion Organiser and the Cambridge Computer Z88.

The Z88 is not, strictly speaking, a pocket organiser as it is the size of an A4 pad. It is less than an inch thick, however, weighs only a couple of pounds and fits snugly into a briefcase or an easily carried soft leather bag.

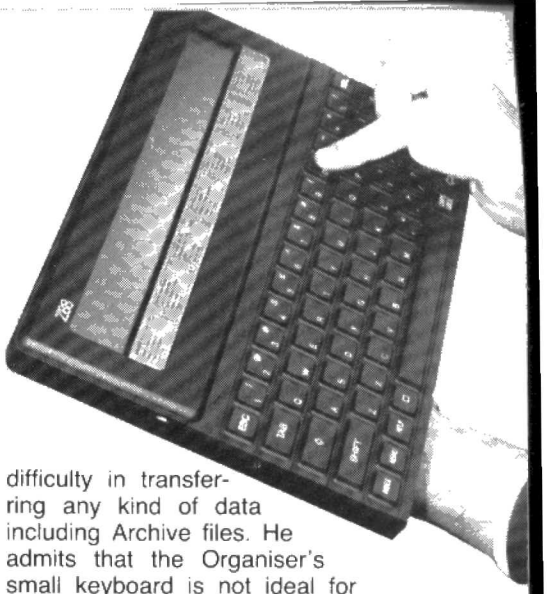
To make transfers to and from the Z88 a special cable and software, costing £25 from Sector Software, is needed. This will convert any files, including Archive, from the QL to the Z88 'Pipedream' system and back again.

David Batty, of Sector Software, considers that the advantages of the full sized keyboard on the Z88 more than compensates for its larger size, particularly for people who want to input data away from home for transfer to the QL.

The machine also has all the facilities of an electronic pocket organiser including a diary, calendar and clock and an alarm system to give reminders of things to do.

The ever popular Psion Organiser, which now comes in several versions, also needs a comms link system if it is to be used with the QL. This pack, the Psion Comms Link, costs £60 and can be obtained from TK Computerware. An ordinary serial printer lead can be used to connect the two units. By itself the link is not very user friendly as all the files would have to be transferred manually. However, the Qualsoft *File Transfer* program, obtainable from TF Services for £7.50, will solve this problem. Also, if a serial printer lead is not available TF services can supply one for £10.

Tony Firshman, of TF Services, uses a Psion Organiser himself and finds no difficulty in transferring any kind of data including Archive files. He admits that the Organiser's small keyboard is not ideal for inputting larger quantities of data but, otherwise, the machine is in every sense a full-blown computer.



Again, Tony finds that the information most worth transferring from the QL is the address book and telephone directory

So there we have it, the Psion Organiser, the Z88 or just good, old fashioned paper. Interestingly, more manufacturers are now bringing electronic organisers into the market, some with the facility to "dial" telephones directly by use of an inbuilt tonepad. Whether it is worth waiting to see whether any QL comms links and programs come on the market to support them is anybody's guess.

DIY TOOLKIT

This month Simon Goodwin premieres four new functions for SuperBASIC programmers even for beginners.

This column introduces a quartet of SuperBASIC functions: NEWCHAN%, LOOKUP%, UPPER% and LOWER%. NEWCHAN% is derived from a Microsoft QuickBasic function, while UPPER$ and LOWER$ are improvements upon Psion *Archive*. LOOKUP% is a QL-specific goodie with stacks of uses that I shall explain later.

As usual, *DIY Toolkit* reveals how the functions are used, exactly how they work, and when they come in useful. This month's code commentary starts from first principles, so you may find it interesting even if you are a beginner when it comes to QL machine code.

All four functions have been carefully tested with interpreted and compiled SuperBASIC on QLs and Thor XVI. They require just 364 bytes of memory, anywhere in ram or rom, and can be shared between any number of tasks.

```
* QL WORLD DIY TOOLKIT - Useful SuperBASIC functions
* Version 1.2, Copyright 1989 & 1990, Simon N Goodwin.
*
start       lea.l     define,a1     Point A1 at definition
            move.w    $110,a2       BP.INIT vector
            jmp       (a2)          Link code to SuperBASIC
*
* LOOKUP starts by fetching a single string parameter
*
lookup      move.w    $116,a0       Fetch CA.GTSTR vector
            jsr       (a0)          Fetch string parameter
            bne.s     exit          Exit unless it worked
            subq.w    #1,d3         Check for ONE parameter
            bne.s     bad_param     Moan if D3 was not 1
            move.w    0(a1,a6.l),d0
            beq.s     bad_param     Reject a null parameter
            move.w    d0,d5         Save length for later
            lea.l     2(a1),a5      Save offset of text
*
* Equate UPPER/lower case; set bit 5 of parameter bytes
*
            moveq     #32,d7        Case conversion mask
            moveq     #1,d2         Odd/even length mask
            and.l     d0,d2         D2=1 if length is odd
lock_case   or.b      d7,2(a1,a6.l)
            addq.l    #1,a1         Advance through text
            subq.w    #1,d0         Count down length
            bne.s     lock_case     Convert all characters
            adda.l    d2,a1         Skip odd filler byte?
*
* Now 0(A1,A6.L) -> integer result slot, look up name
*
            move.l    a1,a4         Protect A1 from MT.JINF
            moveq     #0,d2         Search entire task tree
            moveq     #0,d1         Look for SuperBASIC
            moveq     #2,d0         MT.JINF Trap key
            trap      #1            A0 := base of task 0,0
            move.l    a4,a1         Restore A1 for later
            move.l    a0,a2         A2 -> BASIC system vars
            move.l    24(a2),a0     A0 -> Name Table Start
            move.l    28(a2),d0     D0 -> Name Table End
            move.l    32(a2),d3     D3 -> Name List Start
next_name   move.w    2(a0,a2.l),a3  A3.L is offset in NL
            adda.l    d3,a3         (A3,A2.L) -> Name
            cmp.b     0(a3,a2.l),d5  Compare length
            beq.s     got_length    Length matches!
advance_nl  addq.l    #8,a0         Advance through NL
            cmp.l     a0,d0         Stop at the end
            bhi.s     next_name
            moveq     #-7,d4        Signal NOT FOUND
            bra.s     ret_d4
*
bad_param   moveq     #-15,d0       ERR.BP report code
exit        rts                     Return error code in D0
*
```

Newchan%

NEWCHAN% is a function which tells programmers the next unused SuperBASIC channel number — in other words, the next channel number to use. NEWCHAN% helps to keep programs clear and reliable. SuperBasic channels are numbered from zero; normally at least three channels are open, numbered 0, 1 and 2, so PRINT NEWCHAN% returns 3 if no other channels are open, because #3 is the next unused channel number.

BASIC allows numbers up to 32,767 but in practice it is best to use the lowest values possible, as Basic allocates 40 bytes for each possibility — so OPEN #802, ser would gobble up 800 * 40 = 32000 bytes, most of it recording that channels 3 to 801 are unused. If you call NEWCHAN% instead and it returns 3, you save 31960 bytes.

```
* Check the name to see if it matches the parameter
*
got_length move.b   1(a3,a2.l),d4
           or.b     d7,d4          Ensure consistent case
           cmp.b    0(a5,a6.l),d4
           bne.s    advance_nl     Mismatch, try another
           move.w   d5,d6          Save residual length
           subq.w   #2,d6          DBRA count the rest
           bmi.s    found_it       Found name, length 1
           move.l   a5,a4          D4 & A4 are temporaries
check_name move.b   2(a3,a2.l),d4
           or.b     d7,d4          Convert case of name
           addq.l   #1,a3          Step through Name List
           addq.l   #1,a4          Step through parameter
           cmp.b    0(a4,a6.l),d4
           dbne     d6,check_name
           bne.s    advance_nl     Name mismatch, go on
found_it   sub.l    24(a2),a0      A0 := A0 - (BV.NLBAS)
           move.l   a0,d4
           lsr.l    #3,d4          D4 := Name Offset DIV 8
           bra.s    ret_d4
*
* NEWCHAN% scans the SuperBASIC channel table
*
newchan    moveq    #0,d4          Default NEWCHAN% = 0
           move.l   48(a6),a2      A2 := Chan table offset
           move.l   52(a6),a3      A3 := Chan end offset
*
try_chnum  cmpa.l   a3,a2          End of table?
           bcc.s    got_chnum      If so, D4 is NEWCHAN%
           tst.b    0(a2,a6.l)     Un-used entry?
           bmi.s    got_chnum      If so, D4 is NEWCHAN%
           addq.w   #1,d4          Count one more in use
           lea.l    40(a2),a2      Advance to next channel
           bra.s    try_chnum      Keep trying
*
got_chnum  moveq    #2,d1          Allow 2 bytes for result
           move.w   $11A,a2        Fetch BV.CHRIX vector
           jsr      (a2)           Reserve space
           move.l   $58(a6),a1     Get BV.RIP
           subq.l   #2,a1          Make room for an integer
ret_d4     move.w   d4,0(a1,a6.l)
           move.l   a1,$58(a6)     Save new BV.RIP
ret_int    moveq    #3,d4          Indicate integer result
           moveq    #0,d0          Indicate no error
           rts
*
* Most remaining code is shared by UPPER$ and LOWER$
*
lower      moveq    #0,d7          Set lower flag
getstr     move.w   $116,a0        Fetch CA.GTSTR vector
           jsr      (a0)           Fetch string parameter
           bne.s    exit           Exit if error found
           subq.w   #1,d3          Check for ONE parameter
           bne.s    bad_param      Complain unless D3 was 1
           move.w   0(a1,a6.l),d0  D0 := Text length
           beq.s    ret_string     Zero length is trivial
           subq.w   #1,d0          Adjust length for DBRA
           move.l   a1,a2          A2 is moving text ptr
           tst.b    d7             Check entry flag - is
           beq.s    lower_set      this UPPER$ or LOWER$?
*
* Adjust GROUP 2 values for Greek & Russian ROMs, etc.
*
upper_set  moveq    #97,d2         Start of lower Group 1
           moveq    #122,d3        End of lower Group 1
           move.w   #128,d4        Start of lower Group 2
           move.w   #139,d5        End of lower Group 2
           bra.s    scanner
lower_set  moveq    #65,d2         Start of CAPS Group 1
```

Of course that's a contrived example, and you are more likely to find that NEWCHAN% saves you a few bytes, while helping to make your programs reliable. If your Basic is plastered with literal references to small integers like #4, #6 or #9 it's easy to make a small mistake and direct part of your data to the wrong device.

The more complex your program, or the more channels you use, the more likely it is that you'll mis-type a value or USE the wrong default (March 1988). These bugs can be hard to find, and sometimes hard to fix.

They're less likely if you call NEWCHAN% whenever you want to open a new channel, and store the result in a variable with a meaningful name. Of course you could dispense with NEWCHAN% and set the variable to a constant, but that increases the risk of confusion or wasted memory if the program gets more complicated or you decide to re-use a routine.

NEWCHAN% is especially useful in compiled programs, as high channel numbers can crash tasks. SuperBasic compilers allocate preset space for channel details. Compiled programs cannot dynamically expand that space, so its's important to make the best possible use of available channels.

You may hit problems in compiled tasks if you use more than the maximum number of channels (often 16) at once. In practice most errors are caused by needlessly over-sized channel numbers, like #100, used after #3. NEWCHAN% ensures that you use all the channels you're allowed, first.

NEWCHAN% was suggested by *Taskforce* author Phil Spink, who reports that Microsoft's QuickBASIC has a similar function.

Case Study

UPPER$ and LOWER$ work like their namesakes in *Archive*, but recognise accented characters and work much more quickly. They can be configured to work with non-Roman alphabets, like Greek and Russian, as befits the polylingual QL.

UPPER$ takes a string and converts the small letters into capitals (known to printers as 'upper case'). LOWER$ does the reverse, replacing capitals with corresponding 'lower case'

UPPER$ and LOWER$ are very fast. They ran 5.5 times faster than their Archive equivalents, when I compared Archive 2.38 with UPPER$ and LOWER$ on a QL with "MG" SuperBasic. Of course, much of the time is taken up by interpretation.

UPPER$ and LOWER$ are even faster in compiled Basic; sadly, you cannot compile Archive programs. Years ago Psion's Dick Harrison discussed the idea of writing an Archive compiler with Chas Dillon; they concluded that it would be a

```
            moveq     #90,d3        End of CAPS Group 1
            move.w    #160,d4       Start of CAPS Group 2
            move.w    #171,d5       End of CAPS Group 2
*
* D2 is Start of target group 1, D3 is End of Group 1.
* In group 2, D4 is Start, D5 is End. D2 < D3 < D4 < D5
*
scanner     moveq     #32,d7        Prepare conversion mask
convert     move.b    2(a2,a6.l),d6
            cmp.b     d2,d6
            bcs.s     advance       Ignore all codes < D2
            cmp.b     d3,d6
            bls.s     tweak_case    Tweak Group 1 code
try_group2  cmp.b     d4,d6         Try Group 2
            bcs.s     advance       Ignore if code < D4
            cmp.b     d5,d6
            bhi.s     advance       Too high, no tweak needed
tweak_case  eor.b     d7,2(a2,a6.l)
advance     addq.l    #1,a2         Advance through parameter
            dbra      d0,convert
ret_string  moveq     #1,d4         String stacked at (A6,A1)
            moveq     #0,d0
            rts
*
upper       moveq     #1,d7         Set UPPER CASE flag
            bra.s     getstr
*
define      dc.w      0,0           No procedures
            dc.w      4             Four functions
            dc.w      newchan-*
            dc.b      8,'NEWCHAN%'
            dc.w      lookup-*
            dc.b      7,'LOOKUP%'
            dc.w      upper-*
            dc.b      6,'UPPER$'
            dc.w      lower-*
            dc.b      6,'LOWER$'
            dc.w      0
            end
```

lot of work for very little return.

The tests used strings of 100 characters. Archive cannot handle strings beyond 255 characters, while UPPER$ and LOWER$ take up to 32K in their stride.

Listing 3 is a simple demonstration of UPPER$, LOWER$ and NEWCHAN%. The program displays a file, converting all text to upper or lower case. You choose the file name and press C for capitals or L for Lower Case. Use ram disk to see the true speed of UPPER$ and LOWER$.

Lookup%

LOOKUP% has a myriad uses. You tell it a name from a program, and it tells you what SuperBasic knows about that name. In fact it returns the index number of the supplied name in the SuperBasic Name Table, which is the key to the system.

LOOKUP% takes one parameter — either a string variable, a string expression or a name in quotes — and looks it up in SuperBasic's internal tables. SuperBasic uses names for variables, procedures, functions, files, devices, loops and arrays. The Name Table keeps all these names distinct.

LOOKUP% can tell you the DEF line for any name, or the type, be it a variable, FOR or REPeat loop name, array, string, integer or decimal value. It can reveal the DEF line of PROCedures and FuNctions, and the code address for all Toolkit and rom commands.

Tasks can LOOKUP% a name and find its Basic value. You can change values, if you're feeling brave, using LOOKUP% and BPOKE. You can even change the name, although it's hard to change the length without creating a new variable. All the required information is in past DIY Toolkit articles or Jan Jones' *SuperBasic Handbook*, reprinted by Quanta.

LOOKUP% returns a whole number to indicate how it got on. You get -7 (the Qdos code for *not found*) if the name is not defined; otherwise the result is the index of the name in the Name Table. The first entry in the list (usually PRINT) has index zero.

I wrote BASIC_INDEX%, a similar function, for Turbo Toolkit in 1986. LOOKUP% finds names faster, and ignores case; LOOKUP% ("SER") is equivalent to LOOKUP% ("ser"). SuperBasic ignores case when comparing names, so it is important that LOOKUP% does the same.

It is unwise for programs to assume that names will be in capitals. If you try to use a Toolkit command before it is loaded you get a 'bad name' error. If you load the toolkit code and try again, you find that the command works, but the name is SAVEd and listed with the same capitalisation you used when it was rejected. This is because SuperBasic only stores names once, when it first encounters them.

If you get names right first time, this elegant feature makes it easy to spot typing mistakes later — but sometimes it gets out of control. It is hard to change the case of a toolkit name once loaded, especially as LOAD and NEW do not affect resident procedures and functions.

Cosmetic problems crop up if you type a Basic name the wrong way the first time you enter it. You're stuck with the original mix of capitals and lower case every time the name appears.

You could use DIY Toolkit's REPLACE command (June 1989), but that has limitations; it won't work with procedure or function names, and requires you to choose a different name, not just one with a different mixture of capitals and lower case.

You might SAVE the program, replace the first instance of the name in a text editor, and re-load. This is slow, but it works for all but resident names.

Some programmers favour a combination of SAVE and MERGE. First SAVE, then type NEW. Now enter all the names back the way you want them, with a dummy line like this:

```
1 NAME1=NAME2=NAME3 etc.
```

Just entering all the names in any good line will teach the new capitalisation to Basic. Now re-load your program, using MERGE instead of LOAD, so that the re-defined names persist. Finally, scrub the dummy line, unless your program had loaded over it. All corresponding names in the program will use the fixed capitalisation.

You can use the LOOKUP% to find out what Toolkits are loaded, or to check for other resources which come with Toolkit commands. Look for FSERVE if you need a file server, SPEED if you need Speedscreen, and so on. BOOT programs can check for common extensions like WCOPY and FLP USE before they try to use them.

Armed with BPEEK and BPOKE (DIY Toolkit December 1988) you can write a routine to change the case of a name in situ, in the Name List. BPEEK L (32) is the offset of the start of the Name List. Likewise BPEEK L (24) points at the Name Table, which holds eight bytes for each name. Bytes 3 and 4 are the offset of the Name in the Name List.

This gives the address offset of the first byte of the name "LOOKUP%" in the Name List:

```
Offset                =BPEEK_L(32) +
BPEEK_W%(BPEEK_L(24) +
```

LOOKUP% ("LOOKUP%") *8+2)

You can change the name with BPOKE, but should avoid characters that would not normally be allowed, and remember that the first byte of the name is the length, and should be left alone.

The code for the functions is listed in two forms. **Listing 2** is a quick way to enter the code without using an assembler. It loads the equivalent machine code from DATA statements, and saves the code in a file. Once you've loaded that file, as follows, you can use UPPER$, LOWER$, NEWCHAN% and LOOKUP% in your own programs.

base=RESPR (364)
LBYTES "filename", base
CALL base: NEW

The first part of **Listing 2** is the standard loader used in every month's DIY Toolkit project. Only the DATA, from line 590 onwards, change from month to month.

Listing 1is the corresponding assembly code program, written and assembled using HiSoft's *DevPac*. Type this text into your own assembler if you want to customise the code.

If you are new to 68008 machine code, the NEWCHAN% routine is a good place to start to learn. The code for NEWCHAN% begins about half way through **Listing 1**, after the comment' NEWCHAN% scans the SuperBASIC channel table'. Comment lines are easy to spot because they start with asterisks.

The first line of code starts 'newchan'. Names in the first column of an assembler listing have much the same effect as line-numbers — they serve to identify the line, so that other part of the program can refer to it by a 'label'.

The next two columns of the same line indicate the machine code to be stored at the point. Column Two is the 'opcode', the instruction itself. Column Three holds the 'operand' or 'operands' the registers or values used by the instruction. MOVEQ #O, D4 moves the value zero into register D4, a store inside the processor.

MOVEQ is a quick version of the general-purpose MOVE instruction. MOVE #O, DO is slower. In either case, the hash before the zero indicates that the instruction refers to the *value* zero rather than the contents of address zero. MOVE O,DO copies *the value at address zero* into DO.

The last column is reserved for comments. These are optional, like the label, but help to make the program easier for humans to understand. Assembly code may be written in CAPITALS, but I prefer lower case, because ascenders and descenders make it easier to tell letters apart. Digits in old books of mathematical tablets often have ascenders and descenders for just this reason. There must be at least one space between columns, and no spaces in any label, opcode or operand, or the assembler may reject the line. Some assemblers expect a semi-colon at the start of each comment, but that's a throwback to older languages, and should not be necessary on the 68008.

The QL may have to re-locate SuperBasic as tasks and devices grab memory, so it keeps the current start address of Basic in register A6. A6 changes whenever Basic moves. Programs can still find all the information about SuperBasic, as long as they use the latest value of A6 to find the data. The next line reads a value from Basic. Move. L indicates that a four byte 'long word' value is to be read. The operand 48(A6), A2 means that the

```
100 REMark Sinclair QL World HEX LOADER v 3
110 REMark by Marcus Jeffery & Simon N Goodwin
120 :
150 CLS: RESTORE : READ space: start=ALCHP(space)
160 PRINT "Loading Hex..." : HEX_LOAD start
170 INPUT "Save to file...";f$
180 SBYTES f$,start,byte : STOP
190 :
200 DEFine FuNction DECIMAL(x)
210 RETurn CODE(h$(x))-48-7*(h$(x)>"9")
220 END DEFine DECIMAL
230 :
240 DEFine PROCedure HEX_LOAD(start)
290 byte = 0 : checksum = 0
300 REPeat load_hex_digits
310    READ h$
320    IF h$="*" : EXIT load_hex_digits
330    IF LEN(h$) MOD 2
340       PRINT"Odd number of hex digits in: ";h$
350       STOP
360    END IF
370    FOR b = 1 TO LEN(h$) STEP 2
380       hb = DECIMAL(b) : lb = DECIMAL(b+1)
390       IF hb<0 OR hb>15 OR lb<0 OR lb>15
400          PRINT"Illegal hex digit in: ";h$ : STOP
420       END IF
430       POKE start+byte,16*hb+lb
440       checksum = checksum + 16*hb + lb
450       byte = byte + 1
460    END FOR b
470 END REPeat load_hex_digits
480 READ check
490 IF check <> checksum
500    PRINT"Checksum incorrect. Recheck data.":STOP
520 END IF
530 PRINT"Checksum correct, data entered at: ";start
560 END DEFine HEX_LOAD
570 :
580 REMark Space requirements for the machine code
590 DATA 364
600 :
610 REMark Machine code data
620 DATA "43FA013834790000","01104ED230790000"
630 DATA "01164E9066545343","664E3031E8006748"
640 DATA "3A004BE900027E20","7401C4808F31E802"
650 DATA "5289534066F6D3C2","2849740072007002"
660 DATA "4E41224C2448206A","0018202A001C262A"
670 DATA "00203670A802D7C3","BA33A800670E5088"
680 DATA "B08862EE78F96062","70F14E751833A801"
690 DATA "8807B835E80066E6","3C0555466B16284D"
700 DATA "1833A8028807528B","528CB834E80056CE"
710 DATA "FFF066CA91EA0018","2808E68C602C7800"
720 DATA "246E0030266E0034","B5CB640E4A32E800"
730 DATA "6B08524445EA0028","60EE720234790000"
740 DATA "011A4E92226E0058","55893384E8002D49"
750 DATA "0058780370004E75","7E00307900000116"
760 DATA "4E90668653436680","3031E80067425340"
770 DATA "24494A07670E7461","767A383C00803A3C"
780 DATA "008B600C7441765A","383C00A03A3C00AB"
790 DATA "7E201C32E802BC02","6510BC036308BC04"
800 DATA "6508BC056204BF32","E802528A51C8FFE4"
810 DATA "780170004E757E01","60A0000000000004"
820 DATA "FF5E084E45574348","414E2500FEC0074C"
830 DATA "4F4F4B555025FFE0","0655505045522400"
840 DATA "FF78064C4F574552","24000000",'*',30265
```

```
100 DIM file$(42),line$(64),case$(1)
110 windo%=NEWCHAN%
120 OPEN #windo%,con_512x256a0x0
130 CSIZE #windo%,1,0 : INK #windo%,7 : CLS #windo%
140 REPeat poll
150   INPUT #windo%,"Enter file name to view : ";file$
160   IF file$="" : EXIT poll
170   file%=NEWCHAN%
180   OPEN_IN #file%,file$
190   INPUT #windo%,"CAPS or lower case (C/L): ";case$
200   IF case$="" : NEXT poll
210   REPeat lines
220     IF EOF(#file%) : EXIT lines
230     INPUT #file%,line$
240     IF case$(1)=="c"
250       PRINT #windo%,UPPER$(line$)
260     ELSE
270       PRINT #windo%,LOWER$(line$)
280     END IF
290   END REPeat lines
300   CLOSE #file%
310 END REPeat poll
320 CLOSE #windo% : STOP
```

value at the address 48 bytes on from the address in register A6 must be copied into register A2.

SuperBasic storage starts with a table of address offsets, as I have noted in past articles. These offsets are used to find other tables. The system stores offsets in the SuperBasic area, rather than addresses, so it does not have to update them whenever Basic moves. Each value is a long word; 48(A6) gives the offset which indicates the address of the Channel Table.

The third line reads the address of the end of the table from the next long word, at offset 52. This ends up in register A3, so that table is between offset A2 and offset A3, NEWCHAN% works by scanning this table, looking for an unused entry.

The label TRY_CHNUM marks the start of a loop that tests each channel entry in turn. First we compare the values in A3 and A2 in case the Channel Table is empty. The opcode CMPA.L compares two address registers. Good assemblers accept CMP.L, seeing that the operands are address registers.

The CMP instruction does not affect the operands, but leaves its result in 'flags' accessible to later instructions. If A2 is less than A3 the 'carry flag' is 'set' inside the processor.

The next line reads this flag. BCC stands for Branch if Carry Clear the computer will 'branch' to a labelled line if the condition is true. '.S' indicates that this is a 'Short Branch' to an instruction no more than 128 bytes away.

Long branch instructions use twice as many bytes of memory but can jump to 32K. Some assemblers guess if you do not specify a branch size; others assume '.L'.

If we have reached the end of the table the flag is clear and execution continues at label GOT_CHNUM. Othrwise we examine the start of the Table entry. TST.B O(A2, A6.L) indicates that we want to Test the first Byte (offset 0) in the Channel Table (A2) of BASIC (A6). The .L is a precaution explained in my September 1988 column.

The next line tests to see if that byte was negative — a sure sign of an un-used channel entry. BMI stands for 'Branch if Minus'. Igf so, we've found an entry and continue at GOT_CHNUM. Otherwise we add one to the count in D4 (ADDQ.W #1, D4) and advance to the next channel.

Each channel entry uses 40 bytes. LEA.L 40 (A2), A2 loads the 'effective address' 40 bytes beyond A2, into A2. This is the equivalent of ADD #40, A2 but the LEA instruction is faster. The last part of the loop is an unconditional Short BRAnch back to TRY_CHNUM, to test the next entry.

If we reach GOT_CHNUM we know that D4 contains the number of the first un-used entry, or we have checked all the entires and there is none free; either way, D4 holds the lowest unused channel number.

The second half of NEWCHAN% returns the value in D4 to SuperBasic. The code to do this is in the rom, so there's little point duplicating it. Instead we use the routine Sinclair call BV. CHRIX, which reserves space for a result. An integer needs two bytes, so we indicate the amount of room required by moving two into register D1.

The address of routines like BV. CHRIX may vary between rom versions, so Sinclair put a table of useful addresses are known as 'vectors', because they tell you where to go, rather like navigational vectors.

MOVE.W $11A,A2 reads the value at address 282 into register A2. The dollar sign before 11A indicates that it is a number written in base 16, or hexadecimal. You could just as well write MOVE. W 282,A2, but I put $11A as most QL books follow Sinclair's lead and list vector addresses in hex. Don't let it faze you.

Once A2 contains the address of the BV. CHRIX routine. JSR (A2) indicates that the address is in register A2. JSR works like GOSUB in Basic; if all goes well, execution continues on the next line once the subroutine has done its stuff.

BV.CHRIX allocates space in another Basic area, the RI stack. RI stands for Real and Integer — two types of number. Anyhow, we find that area by looking at $58(A6), and copy its offset into A1. We subtract two from A1, to make room for the result,and copy the word in D4 into the space addressed by A1 and A6, before saving the new stack pointer.

The last three lines load DO with zero, indicating no error, and D4 with 3, to signify an integer result, before returning to the main program with RTS.

Lower and Upper

I must explain LOWER$ and UPPER$ in less detail, or I shall run out of room! Much the same code is used for each function; the only differences are the character codes that get change. LOWER$ starts with zero in D7, while UPPER$ loads one; they then share code to fetch a single string parameter.

If the string length is zero there's no need for scanning. Otherwise we must look through the whole string, checking the code of each character and converting it if need be. DO is used as the character count, while A2 holds the offset of the next character.

Four data registers indicate which codes need to be changed. D2 and D3 hold the start and end of the first range of un-accented characters, while D4 and D5 record the second range. The values vary depending on whether we're looking for CAPITALS or lower case. They also depend on the nationality of your QL or Thor roms.

Most QLs have lower-case accents in codes 128 to 139, and equivalent capitals at 160-171. This is true for "AH", "JM", "JS" and most "MG" roms. However languages like Greek and Russian need lots of extra characters. The Greek alphabet uses codes from 128 to 151 for capitals, and codes 160-183 for 24 corresponding lower-case letters. I often use a Greek rom as it was perhaps the best produced by Sinclair.

The Thor XVI contains twelve different language settings, including 64 letters aimed at the Soviet market. The Thor assigns codes 128-159 to Cyrillic capitals, and 160-191 to lower case. UPPER$ and LOWER$ can be tweaked to work perfectly anywhere in the world. Just patch the codes in UPPER SET and LOWER SET to match your own characters.

You are allowed two sets up to 32 characters, with a constant difference of 32 between the codes for each case. The difference of 32 is necessary because the functions convert case by changing the value of bit 5 in each byte, as in *Multibasic*. The first group specified must have codes lower than the second group.

The core of UPPER$ and LOWER$ is the routine labelled SCANNER. This zooms through the string testing each byte to see if it falls into one of the two groups, and changing it if need be.

The loop is written to minimise the number of tests on each character. UPPER$ lets through most capitals and punctuation with just five instructions for each character. UPPER$ and LOWER$ identify and convert each un-accented letter in just eight instructions.

The fearsome EOR.B D7, 2(A2,A6.L) toggles the case of the letter from capitals to lower case, or vice versa by inverting bit 5; this works for Russian, Greek, American and European accents. The program leaves the value on the stack, where it can easily be returned to Basic.

The table labelled DEFINE indicates the names and addresses of the commands. The names are at the end to make it easier to alter them, possibly changing the length of the file.

Back at the top of the listing, the START routine points A1 at the table, and calls BP.INIT, the rom vector that adds new functions to SuperBASIC. This routine is as the start so it's easy to find after loading the code. It is not used later.

LOOKUP% starts by fetching a string parameter onto the Maths Stack, addressed by O(A1,A6.L). Name parameters must be in quotes, or you get an 'error in expression' report. After CA.GTSTR has finished the word at the top of the stack is the length of the string, followed by the characters.

Next, LOOKUP% sets bit 5 in every character of the parameter — this ensures consistent case later, when comparing the parameter with stored names. We began by pointing A1 past the string length, with an offset of 2, so once we have corrected for the 'odd' byte at the end of some strings A1 is the offset of the two byte space on the stack where the integer result will go.

We save A1 while looking for the base address of SuperBasic; we can't just use A6 as LOOKUP% may be used in a compiled task with no Name Table of its own. MT.JINF returns the address of task [0,0] in A0; we copy it to A2, and set registers to point to the Name Table and Name List. Fast code uses registers as much as possible. This is where human programmers excel over compilers, and values in registers can be accessed much faster than memory.

The second word in each eight byte Name Table entry is the offset of the name text in the Name List. LOOKUP% adds D3, the offset of the start of the List, to find the length of the name. If the length is wrong the loop advances immediately to the next name till all have been checked. If no match is found LOOKUP% returns the error code -7.

We only reach GOT LENGTH if the length was right. We want to exclude mismatches as quickly as possible, so we read the first byte of the name and OR it with D7 so its case matches the parameter bytes. If the comparison fails we step straight on to the next name.

The final test loop is similar to the one used in MultiBasic, and not especially fast, but it is only used in rare cases where the length and the first letter match the parameter, so LOOKUP% is fast overall. It compares about 3600 names a second on my QL.

LOOKUP% is quite unlike BASIC INDEX%, which starts by scanning the Name List and only looks up the name in the Name Table if it finds a plausible offset. BASIC INDEX%, is quickest if the name is not in the List, but LOOKUP% is faster if the name is found, despite the need for case conversions.

I have received dozens of letters in response to my plea for new Toolkit ideas; thanks to everyone who sent in suggestions. DIY Toolkit will be back next month, with nifty code, advice and exciting news. Don't miss it!

DOTS TO PAPER

Paul Walton goes back to basics on printing screen dumps with dot matrix printers.

Most modern dot matrix printers offer a wide range of print options from a variety of fonts to dot graphics. Unfortunately many people are unable to exploit these facilities because they cannot make sense of the printer manual – which is not always the reader's fault. Often the description in the manual is full of jargon that assumes a considerable prior knowledge. One technique that frustrates many people is how to use the dot graphics mode of their printer to perform screen dumps, so below, in simple terms and with as little jargon as possible, is an introduction to screen dumps.

For printing characters, most dot matrix printers are EPSON or IBM-compatible; better still, for those printers capable of it, there is also much commonality when printing dot graphics. Perhaps the biggest difference is whether you change to dot graphics mode by flicking a switch inside the printer or by sending a command from the QL. This description is based upon the Citizen 120-D which is of the latter type. It is typical of most modern dot matrix printers and owners of Brothers, Epsons or Stars should find it equally applicable.

Firstly, you need to understand a little of how a dot matrix printer works. It typically has a print head containing a column of 9 or more pins that are pushed forward under the control of the logic in the printer to impact the ribbon on to the paper. The head is stepped across the paper and the pins strike as it moves thus creating characters as patterns of dots – see figure 1.

To operate a printer, the QL sends it ASCII codes. If ASCII codes are new to you, they are numerical codes that printers use in a look-up table manner – send a 75 to the printer and it responds by printing the letter "K". The code is sent to the printer down your printer cable and can be any number from 0 to 255. The first 128 codes are more or less universally agreed, but the second 128 vary and it is here where EPSON and IBM printers differ.

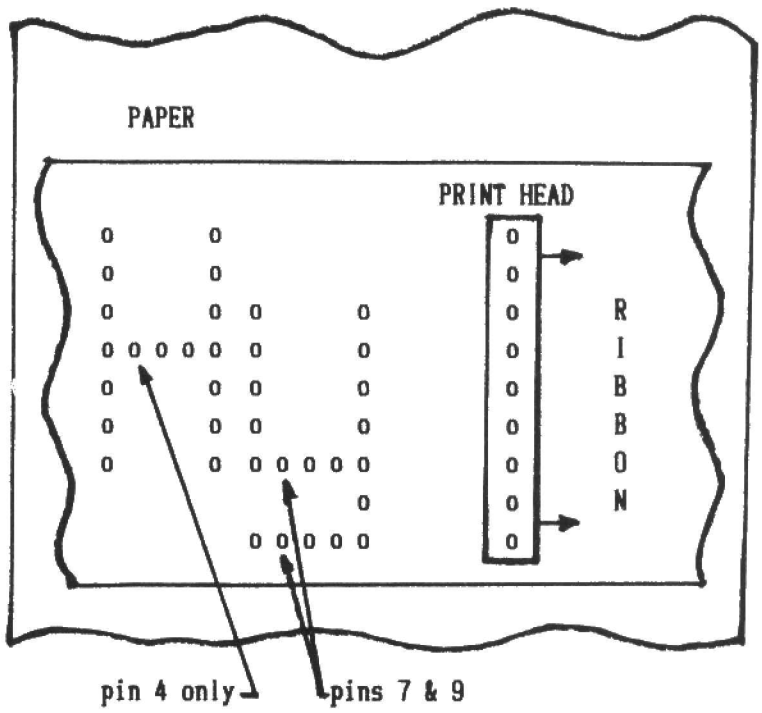


Figure 1. How the print head moves.

128	o	pin 1	128	o
64	o	pin 2	64	o
32	o	pin 3	-	
16	o	pin 4	-	
8	o	pin 5	-	
4	o	pin 6	-	
2	o	pin 7	2	o
1	o	pin 8	1	o
-		pin 9	-	
Figure 2			Figure 3	

Figure 2: The pin binary codes; Figure 3: To fire pins 1,2,7 and 8 the total code is 195.

When printing characters, the printer only needs to be told the ASCII code of the required character and it works out which pins to fire off over the next series of steps to create that character's shape. How far the printer moves in each step and how many steps make a character can vary from one printer to another, but there are typically 6 steps with spacing of 1/60 inch to give 10 characters per inch.

So how are we going to send numbers to the printer that will do screen dumps that may consist not only of characters, but lines and shapes etc. Clearly we need a lot more "codes". Well this is not practical so an ingenious solution is provided by most dot matrix printers. This uses the fact that the code 27 – usually called ESCAPE or ESC for short – does not print a character, instead it is seen by the logic in the printer as meaning that a special sequence of non standard codes is on the way.

For example, on the 120-D, the sequence of codes 27 75 44 1 means "get ready to receive 300 special codes" – the

300 comes from combining the 44 and the 1 as 44+(256*1). The 75 tells it what the horizontal spacing is to be.

Notice that without the 27, the printer would treat the 75 as an ordinary ASCII code and print the character "K". There are a variety of these special sequences, but they have one thing in common – you must get them exactly right for them to work – any error and the printer will ignore them or treat them as ordinary ASCII codes.

Your printer manual may well describe the above sequence as:

CHR$(27) "K" CHR$(44) CHR$(1)

One way to send this sequence to the printer is to open a channel to the serial port and print to that channel:

```
10 OPEN #5, ser1
20 PRINT #5, CHR$(27); "K"; CHR$(44);
CHR$(1);
```

Notice the use of the semicolons to ensure that no extra spaces are sent as this would change the sequence – the printer has to receive the 4 codes with nothing in between. Also notice that sending the code 27 as CHR$(27) results in sending a single 8-bit value of 27 to the printer – it would not work if you used – PRINT #5, 27 – this would send the ASCII code for 2 followed by the code for 7. This is an important point that accounts for much of the difficulty people have in making the printer do what they want.

It does not have to be 300 hundred codes – so your manual may generalise by expressing the sequence as:

CHR$(27) "K" CHR$(n1) CHR$(2)

In this case the number of special codes that are to follow will be determined by the value of n1 + (256 * n2).

Before looking at some other sequences, let us see what happens to the 300 – or whatever – special codes when they are received. I said earlier that the print head is a column of pins. Upon receiving a single ordinary ASCII code, the printer will automatically print 6 or more columns to form the whole character. However, with the special codes, it only prints one column per code – thus you have complete control over the patterns created. A less desirable side effect of this is that it also slows the printing down sixfold!

Typically a 9 pin head will only use the top 8 pins when printing special codes. Remember I said the printer is sent the codes as numbers from 0 to 255 – well these are 8-bit binary numbers. So, for the special codes, each 'bit' in the number of the code fires one of the 8 top pins. If 'bits' and 'binary' are baffling, look at figure 2. The numbers show what is required in the code to fire that pin. To fire several pins in one column, add up the numbers for each and send that as the special code. Thus to fire the top 2 and bottom 2 pins, you send the special code 195 as shown in figure 3.

```
9000 DEFINE PROCEDURE  dump
9010 LOCal first_row, last_row, rows
9020 LOCal first_col, last_col, col_count
9030 LOCal base, addr, n1, n2, i, j
9040 :
9050  first_row = 0: last_row = 255
9060  first_col = 0: last_col = 511
9070  OPEN #5, ser1
9080 :
9090  PRINT #5, CHR$(27); CHR$(65); CHR$(8);
9100  base = 131072 + first_row*80 + first_col
9110  rows = last_row - first_row + 1
9120  n1 = (2 * rows) MOD 256
9130  n2 = (2 * rows) DIV 256
9140 :
9150  col_count = ((last_col - first_col) DIV 4)
9160  FOR i = 0 TO col_count STEP 2
9170    addr = base + i + 128 * rows
9180    PRINT #5, CHR$(27); CHR$(42); CHR$(5);
9190    PRINT #5, CHR$(n1); CHR$(n2);
9200    FOR j = rows TO 1 STEP -1
9210      addr = addr - 128
9220      PRINT #5, CHR$(255 - PEEK(addr));
9230      PRINT #5, CHR$(255 - PEEK(addr+1));
9240    END FOR j
9250    PRINT #5
9260  END FOR i
9270 :
9280  CLOSE #5
9290 END DEFINE dump
```

So what can these special sequences do for us? Here are a few examples of the sort of effects they produce:

27 4 :Selects italic characters.
27 5 :Cancels italic characters.
27 2 :Selects 1/6 inch line spacing.
27 65 10 :Selects 10/72 inch line spacing.
27 76 40 1 :Prints the next 40+256*1 codes at120 dots per inch.

In the above sequences all codes should be sent as – chr$(code); – and in the last example, after the 296 codes have been printed, the printer reverts to normal ASCII.

At last we are ready to write a screen dump program using the special sequences – by the way they are called "escape sequences" because they start with the ESCAPE code, 27.

Escape

To do a screen dump we must make the printer fill the paper both horizontally and vertically with the same pattern of dots that exist on the screen itself. This means that we need to send escape sequences that not only tell it to receive special codes, but also to feed the paper by just the right amount at the end of each line so as to leave no gaps between one row and the next.

The escape sequences required for the 120-D, and many other dot-matrix printers, are:

27 65 8: selects line spacing of 8-72 inch which corresponds to the height of the 8 pins.
27 42 4 0 2: selects 80 dots per inch horizontal spacing and also warns the printer that 512 special codes follow.

One problem is that if we take two bytes from the screen area of the QL memory, they represent – depending on which mode we are in – four or eight horizontal dots of various colours. If these two bytes are sent to the printer, the printer will print two adjacent vertical columns of dots. This 90 degree mis-match is overcome by dumping the display as if your TV or monitor screen was standing on its side. This is the reason for the rather strange loops in the program. They make the procedure send the left hand column of the screen, from bottom to top, as the first row on the paper, the second column as the second row etc.

Colour

Each colour on the QL screen has a "numerical" value and this is sent to the printer where it will fire the appropriate pins. Every time the same colour is sent, the same pins fire and so the pattern on the screen is duplicated at the printers – unfortunately, dark screen colours tend to produce light areas on the paper. The use of (255 – PEEK(addr)) 6 in line 10190 improves the correspondence between screen and paper.

The dump is written in the form of a procedure than can be called from a SUPERBASIC program whenever you want to dump the screen. The variables first_row, last_row, first_col and last_col determine the area of the screen to be dumped and refer to the pixel coordinate system of 512 x 256. The values shown dump the entire screen. The process is not fast – in fact it will be at least six times slower than normal printing which is mostly a function of the printer and not the program.

Distortion

If you use the procedure for dumping regular shapes, such as circles, you will notice one remaining problem – they do not come out circular. The amount of distortion is determined by the ratio between the amount we line feed the paper and the number of dots per inch that we print. Unfortunately, the line feed has to match the vertical extent of the print head while the choice of "dots per inch" on most printers is limited to about eight values.

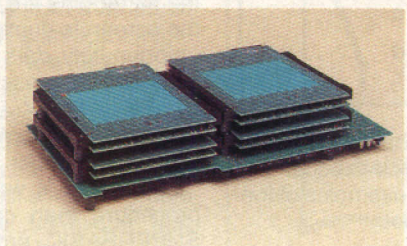
One way around the problem is to choose the dots per inch so that you need less than the normal line feed amount to obtain an undistorted picture. On the 120-D, the line feed spacing can be varied to within 1/216 inch. You then work out how often you should advance by less than the normal amount and on these occasions you only fire seven of the eight pins – or you accept some overprinting of the previous row. As they say in the best books – and this is left as an exercise to the reader – happy dumping.

SINCLAIR WORLD

Bryan Davies meets a distant cousin of the QL at the fair.

It is a pity that the C5 'bicycle' became an object of derision in the UK, because that automatically blackened Sinclair's name too. To my mind, we may all find ourselves using some descendent of the C5 in the not-to distant future, as car travel becomes impossible (particularly in the south-east of England).

In the meantime, an example of the knock-on effect of the C5 saga was evident at this year's "Which Computer?" show. I came across a stand which had on display a "wafer-scale disk drive" and, on enquiry, found that it was indeed from Anamartic, one of Sinclair's latest business ventures.The drive had a capacity of 40MB (although up to 160MB is said to be available) and it was stated to be fully operational, and available now. The price was horrendous — £7,000-odd — and the unit was even larger than the Miracle hard disk unit (also 40MB), but the big thing is that the technology seems now to have reached production.



The wafers are made by Fujitsu, which is not a company to be ignored (number two in the world computer stakes?), and they also have a financial stake in Anamartic. The striking comment from the man on the stand was that they were not mentioning the name Sinclair at all in connection with the drive, because of the aura created around the name by the C5.

In case water-scale integration means nothing to you, it's worth commenting on it, because it looks very possible that it will be the way storage will be produced in the near future. After quite a few years of integrated circuit chip production, the yield from the manufacturing process is still poor. That is, chips are made in quite large blocks — wafers — and only a percentage (said to be 30-60%) of the chips on a wafer are good ones.

The wafers are cut up to produce individual chips, and there are second-grade and maybe lower ones, apart from the good ones. For example, the good ones may work at a 33MHz clock rates, but others which can't manage that can still be sold as 25MHZ or 20MHZ devices. Even so, a lot of chips are scrap. The merit of Sinclair's device is that the whole wafer is used. Each ic on the wafer has a program logic device placed next to it, and these look at the wafer and identify good and bad circuits on it, then manage input/output so that the paths of data within the wafer use only good circuits.

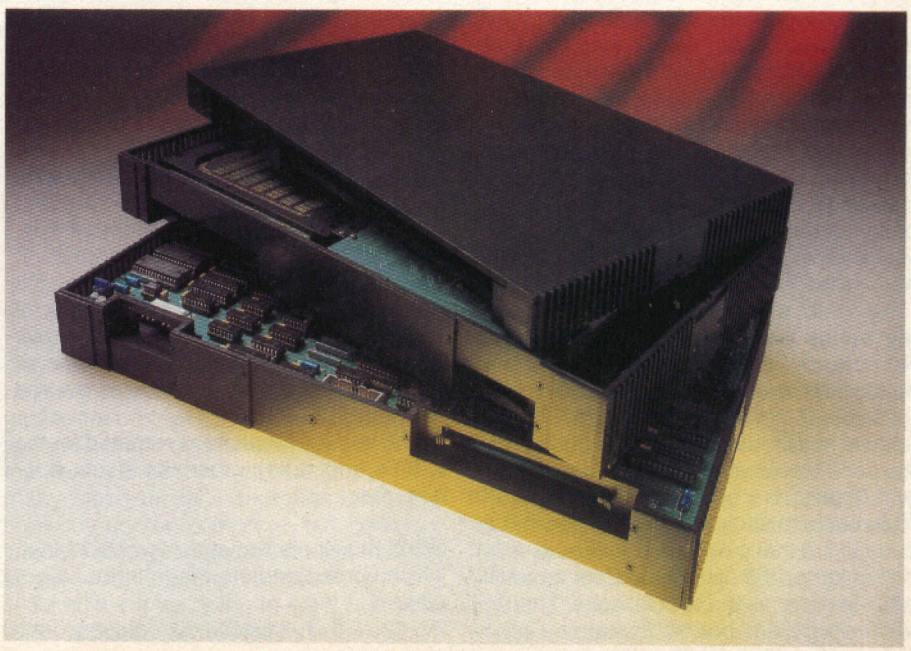
The data path is a spiral, which is not unlike that on a disk. It starts from the outer edge and works into the middle. To keep track of the configuration (the good and bad area) of each wafer, there is an external memory area, which is like a battery-backed clock in that it doesn't lose what is in the memory when the unit is switched off.

There is no bother cutting up the wafers, and then putting each chip into its own case. The process is dynamic, in that failures of chips in service are handled by the software. Presumably the drive goes on working more-or-less "down to the last chip". This is comparable with the process of marking bad blocks on hard disks, to prevent software trying to use them and creating errors, but that has to be done manually when formatting disks, or maintaining them afterwards. The other big point is that the wafer storage is similar to ram memory, being constructed with integrated circuits of a similar nature, and access is very much faster than on mechanical devices such as hard disks. Expect wafer devices to gradually replace hard disks over the next few years.That august establishment, the Pentagon, is quoted as classifying wafer-scale integration as "one of six critical technologies for the 1990s."

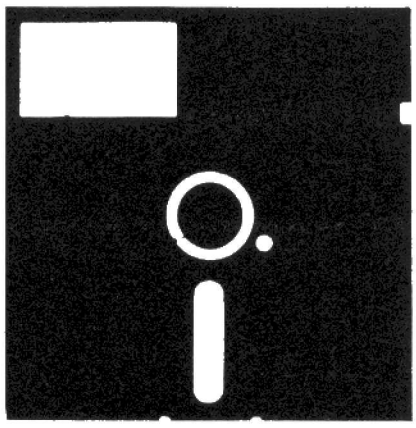
Here are a few more of the claimed advantages of this new technology. More of the chips on a wafer can be used (less are "thrown away" because the performance standard required of them is lower). Most of the wires and soldered joints needed in conventional drives have been done away with. As the chips are not packaged (in plastic) there is little heat-dissipation problem, and fewer "soft errors" (ones which appear now-and-then) occur. 50% more information can be stored in a given physical volume. This latter point sounds sensible, until you see the size of the unit that was on display; it struck me as being remarkably large for the capacity.

The current Fijitsu wafers use standard cmos 1 MB dram chip technology (as on the latest Trump Card) and have a capacity of 20 MB. Without doubt, 4- or even 8- or 16 MB chip versions will be available before too long. One of the reasons for this is that the manufacturers do not have to wait until the yield is high enough for it to be worthwhile selling packaged chips. New chip designs will get used on wafers before they appear in discrete chips.The speed advantage over typical conventional drives is of the order of 100 times. That is, around 0.02ms access time instead of 20ms (which is still faster than hard disks in most home Pcs).

You might well ask whether there is any reason for the technology not to be applied to ram, eprom etc. Apparently not, and it is predicted that entire computer systems will be based upon wafer-scale integration, using Anamartic's technology. Those who are a bit long in the tooth may remember mention of wafer-scale in connection with the QL. It doesn't look as though we will get the benefit of it now, but it's nice to see the suggestion was not simply hot air.



DISK TIPS



The disk revolution is with us — for those who want it. Bryan Davies discusses the different types of disks and takes a brief look at interfaces.

The intention in this article is not to delve deeply into technical matters, but to make a few simple points about the equipment, which may help users with limited technical knowledge and capability to understand it better.

Floppy Disk

"Floppy" derives from the nature of the 5¼in disk, which is not exactly a rigid thing. Like it or not, the name now covers the other small disks you are likely to come across — 3½in and 3in. Presumably, the 2in disks now used in some portable computers are also called floppies. All these latter disks are far from floppy. They have robust plastic casings and can withstand a lot more punishment that 5¼in disks, although, to be fair, I come across little trouble that can be blamed on the packaging with any disk format. There are always people who do the unbelievable, however, and there are regularly reports of 5¼in disks having coffee poured in them, being stapled, and having the inner magnetic disk removed from the protective outer casing. (It may be very hard to believe, but computer journalists assure us that some users who are asked to send copies of disks for examination put photocopies of the disks concerned into the post.)

The 2in disk drive was produced because there is very little space to play with in a portable computer; also, the smaller the unit, the lower the weight, and portables need to be as light as possible. You are not likely to have to bother about the size of some years yet but, when it becomes common, it is likely to store 2MB or more per disk. The 3in disk comes from Sony, and has failed to catch on. The obvious well-known computers using it were the Amstrad PCW models. It is somewhat more cumbersome than the 3½in thicker and with generally less storage capacity; it has nothing going for it now, and you should not need to both about it in future. The 3½in type is the one to take an interest in. It has superseded the 5¼in type as the standard size, although it may never completely oust it because there is such an enormous base of existing 5¼in units. The number of IBM-type computers in existance is said to be around 30,000,000, and most of them will have at least one 5¼in drive, making it reasonably certain this size will be supported by PC manufacturers and software suppliers for many years to come.

If you are considering buying drives for your QL, go for 3½ in. You should be able to obtain any software you want on that format now and, as far as the QL is concerned, you may find some difficulty in getting software on 5¼in disks in future. Several PC software houses are currently supplying their software on both 5¼in and 3½in disks (together), but that won't happen on cheaper QL scene.

There is endless confusion over the nomenclature used to describe disk drive types, and I don't pretend to understand it clearly, but here are a few pointers. The main point to note when comparing QL drives with those on a PC are that the 5¼in type on the QL can be either 40- or 80-track devices, whereas those on the basic PC are always 40-track. The QL 80-track drive should be able to handle 40-track disks also, by ignoring alternate tracks. It may not, however, be able to format a disk to 40 tracks. The 40-track drive will not handle 80-track disks, period. Therefore, it is not advisable to buy 40 track drives for your QL.

If you want to swap 5¼in disks between QL and PC, using programs such as *Conqueror* and *DiscOver,* your QL drive must be able to format disks to 40 tracks, because PCs will generally not accept 80-track disks. There is a possibility of trouble when swapping disks between drives which are of different types, in respect of the number of tracks they write/read to in normal mode. The 40-track PC drive may "see" information between the tracks of 80-track disk which has been written to as 40-track on the QL. The same problem can occur when swapping disks between HD (high density) and DD (double density) drives on PCs, and the way to avoid it is to format the disks only on the lower-density drive, in each case — That is, format your disks to 40 tracks on the PC, before writing to them on the QL.

This problem does not occur with 3½in drives, because they will be 80-track on both types of computer. It is only on more-recent PCs (mainly of the AT type) that 80-track 5¼in drives have been fitted, *and these are not the same as on the QL*, being so-called "high-density" drives and giving 1.2MB storage space; they can write/read to DD 360KB disks, though.

Are you following so far? There are (at least) two further variables to consider if you use the *Conqueror* MS-DOS emulator on the QL. Your 80-track 5¼in QL drive is capable of formatting to 720KB, provided MS-DOS will accept that, and the version 4.01 supplied with Conqueror will do; you do have to configure Conqueror itself first, though, to identify your drives as "what they are not". Being able to get only 360KB on a disk is rather restrictive, and setting Conqueror to allow 720KB on 5¼in disks makes life easier, but don't expect disks formatted this way to be readable in a PC.

FLOPPY DISC

The other obvious odd combination is rather the reverse — to format 3½in disks to 360KB on the QL. So far as I could see, this would be a pointless exercise, since the PC accepted a disk formatted in this way without murmur (it refused to perform such a format, however). The only occasion I used this function (with *Solution* at that time) was when trying to run the Ant "MS-DOS emulator", which required 40-track 3½in. disks, for some unexplained reason. Don't be smart, like me, and try to achieve these odd formats from the Conqueror MS-DOS command line by using what appear to be the appropriate parameters; use the Conqueror *Configure* program to change the perceived types of the drives.

Early PCs had 5¼in drives which used single-sided disks; data was recorded on one side of the disk only, and the storage capacity was 160KB or 180KB. The majority of PCs are fitted with drives which take double-sided disks, and the capacity is 360KB. Two sides, twice the capacity. The PC never went to the next step — 720KB — but jumped straight to 1.2MB (still at 5¼in).

With no more sides available, the increase in capacity had to come from packing the information more densely onto the same area.The 360KB drive is described as double-density, but has only

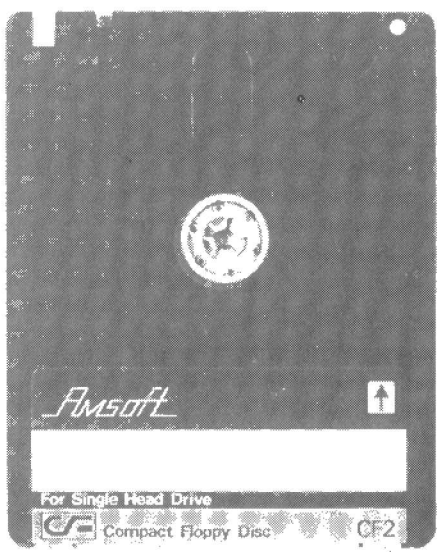
40 tracks and 48 tpi (tracks per inch). The 720KB drive commonly used on the QL is also double-density, but has 80 tracks. There is another classification — Quad-density — which also has 80 tracks and 96 tpi, but that need not concern us here. (I prefer to skip the density classifications, as they refer to the method of recording data onto disk, rather than to the physical arrangement of the disk, and tend to lead to confusion.) The final jump to 1.2MB is to a category called "high-density" although still 80-track. I am uncertain of the tpi figure, which seems to be quoted still at 96, but may be more. This type is for PCs, and is not usable on the QL.

If you're not clear on KB one byte (= 8 bits) of storage space is roughly what is needed for one character (a letter of the alphabet, or a single-digit number) and 1 kilobyte (= 1KB) is about enough space to store 1,000 characters. Strictly-speaking, kilo as used in computers is 2 to the power of 10, or 1024, and its symbol is K (not k). Ram or rom memory inside the computer is usually quoted in sizes such as 1024-, 512-, 256- or 128 KB. For non-technical users, it is can be simpler to forget the odd 24 and say a kilobyte is around 1,000 characters, but the error in doing this gets quite significant whern you start talking about hard disk drives. 30MB drives are sometimes quoted as 32MB drives, apparently because the number of bytes that can be stored starts with 32, but you have an error of about 2 million characters when you call such a drive "32MB", and that's a lot of space (about three times what you can get on one 3½in. floppy). To get a feel for storage space,think of typing an A4 page of notes. You have about 65 lines, with roughly 75 columns for each. That's a possible 65 × 75 = 4875 character spaces. For simplicity, say 5,000. A word may typically require only six characters, but you have to store the spaces and punctuation also. If one A4 page holds 5,000 characters, it will take about 5 KB storage space to hold that pages. Therefore, a 720KB floppy will hold around 720/5 = 144 A4 pages of information. (Compare that to your microdrive cartridge, with capacity for only about 20 full pages).

In practice, many people will put far less than 5,000 characters on a page, and the program being used will be able to record the fact that a page is half-empty without needing to store 2,500 spaces separately. So you may get far more of "your pages" onto the storage medium.

The old PC disk capacity of 180KB is still supported by operating systems such as CP/M and MS-DOS, but the majority of PCs you are likely to see around now use the size which came next — 360KB. This uses a double-sided disk basically the same as the 180KB type, but with data recorded on both sides. Unlike some Amstrad PCW drives, which need the disk to be physically turned over for data to be recorded on both sides, the standard 360KB disk is always inserted the one way only, and the drive mechanism takes care of recording on both sides by having two recording heads, one above and one below the disk. You may find some of your disks are marked single-sided, but this is likely to be only a marketing dodge, the idea being that you can charge more for something marked "double". In practice, although the single-sided drive is just what it says, the single-sided disk can be used in the same way as a double-sided one.

The QL came along a few years after the PC and started out with the capacity to support 720KB disks. Having got to 360KB by using both sides of the disk, another step is needed to double the capacity again, and you may have noted that your disks are often marked both double-sided double-density, and 96 tpi. Alternatively, you may have seen 80 tracks (= 96 tpi) on them, rather than the 40 tracks (= 48 tpi) of the 360KB disk. The magnetic tracks used are closer together,



to allow more information to be recorded. With the 3½in disk, there is much less space to pack the information into, and the tracks have to be spaced even closed together, at 135 tpi; you still get the same 720KB of storage space as on the 96 tpi 5¼in disk, but in less area.

A brief resume seems desirable at this point. The abbreviations commonly used are SS for single-sided, DS for double-sided, SD for single-density, DD for double-density, HD for high-density. 40-track is equivalent to 48 tpi, and 80-track to 96 tpi. All the types we are interested in have each track divided up into nine sectors, so that a DS 80-track disk has 2 × 80 × 9 = 1440 sectors, which — at 512 bytes bytes per sector — equates to 720KB. In contracts, the HD 5¼in drive used on recent PCs divides each track into 15 sectors — 2 × 80 × 18 = 2400 sectors = 1.2MB, and the HD 3½in drive uses 18 sectors, to give 2 × 80 × 18 = 2880 sectors = 1.44 MB. The storage capacities given are formatted space:

SS 48 tpi, 9 sectors gives a capaity of 180 KB.
SS 96 tpi, 9 sectors gives 360 KB.
DS 48 tpi, 9 sectors gives 720 KB
DS HD 96 (?) TPI, 15 sectors gives 1.2 MB.
DS HD 135 tpi, 18 sectors gives 1.44 MB.

There are a variety of identifying markings on disks. A couple of samples for 3½in disks are "Sony Micro Floppydisk, double sided, MFD-2DD" and "3M, 3.5, DS, HD, double sided, high density, 2.0MB". The first identifies a standard 720KB 3½in disk, manufactured by Sony. MFD is presumably their abbreviation for micro floppy disk, and 2DD indicates double-sided and double-density. By implication, this disk has 135 tracks per inch. It could also have been marked "1 MB", because the *unformatted* capacity of such disks is that amount. This is of no great interest to us, since the capacity we can use is the QDOS (or MS-DOS) *formatted capacity,* and that is only 720KB.

The other disk is made by 3M (Minnesota Mining & Manufacturing Company), is also 3½in and double-sided, but is high-density and therefore has an unformatted capacity of 2 MB, which equates to 1.44MB formatted capacity (on a PC). I don't recollect seeing a tpi figure quoted for HD disks; logically, it would be 2 × 135 = 279 tpi, since the disks are the same physical size as DD disks but have twice the storage capacity.

ODD COMBINATION

The situation with 5¼in disks is confusing to say the least. I've just fished out a few samples, which all format to 360KB on a PC, and would presumably give the same on a 40-track QL drive. They are labelled as follows: DS DD 96/48 tpi, DS DD 96 TPI 1.0MB, DS DD 48 tpi, SS DD 48 tpi, DS DD 40 tracks. The ones marked with 96 tpi would be suitable for 80-track QL drives, and give 720KB on them.

It is, perhaps, unfortunate that the QL operating system QDOS refers to disk space in terms of Sectors. The average user is likely to know that each sector will hold 512 bytes of information, and he/she would be better informed by messages given in bytes rather than sectors. 1440 sectors — the full formatted space available on 80-track disks (of either size) — is equivalent to 512 × 1440/1024 = 720 KB. You don't actually get the space when a format is finished, since QDOS uses part of each disk (and cartridge) for information it requires, so the message 1434/1440 sectors means that the disk formatted correctly to 720KB but only 717KB is available for files.

If the first figure is less than 1434, your disk is not giving the space it should do. Reformatting might bring the figure up to the correct amount. As with cartridges, disks which do not give the full space after

formatting should be considered suspect, and not used for valuable files. Unlike cartridges, disks rarely give less than the maximum space; you should consider purchasing a disk-cleaning kit if you see less than 1434 on more than the odd occasion.

Another point to bear in mind is that deleting all the files on a disk will not necessary yield the full number of free sectors, and it may be necessary to reformat the disk to get the maximum. The same comments apply to 40-track 5¼in disks, except that the figures are half those for 80-track, the capacity being 360KB. Think when you see the 1440, because it could mean KB not sectors, and it then applies of high-density disks (2880 *sectors*), which are not really the right type to use in a QL drive. This type of disk can be formatted to give only 720 KB and then used with the QL but, since it costs much more than standard 3½in disks, there's no sense in buying it.

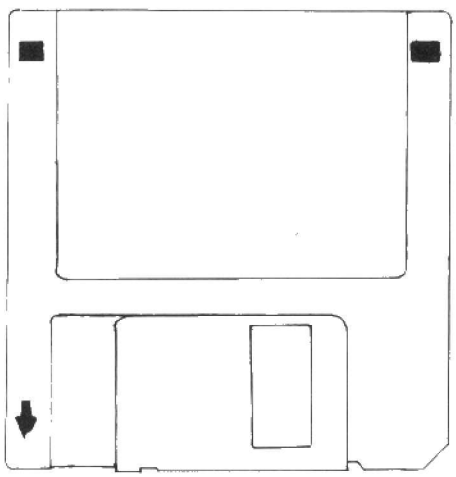
The method of packaging 5¼in disks is such that it is quite easy for "foreign bodies" to get inside the pouch and come into contact with the magnetic disk itself. Part of the disk is always exposed. It is also fairly easy for the whole thing to become distorted, for example through being left in a pile of heavy articles which press unevenly on it. You may have heard a slight scraping sound as the disk goes round; it isn't a healthy sign. The 3½in package is much better, being both more resistant to external pressures and having a sliding panel to cover the magnetic disk when it is not in the drive.

There are various features on each type of disk, which require some explanation. The 5¼in type has the large hole in the middle, the purpose of which is fairly obvious — the drive mechanism fits into the hole in order to rotate the disk. There may or may not be a hub, which is a ring added to the magnetic disk at the centre. It is not an important item for us, as its presence or absence doesn't seem to have any obvious effect, although it must strengthen the disk in the region that will come under most stress. More important is the small hole just outside the central cutout. This is for locating the start of the disk. You may notice when the disk is removed that there is a small hole in the disk itself showing in the pouch hole. This hole is detected optically and enables the drive to know which part of the disk it is looking at.

Another optical detector is aligned with the small square cut out in the one edge of the pouch. The purpose of this is to indicate to the drive whether or not the disk should be written to. If the cutout is not obstructed, the drive knows that the user is allowing the disk concerned to be written to freely. That is, whenever a Save, Copy, Delete etc. operation is commanded, the drive can do what is requested. The disk is in the "unprotected" stage.

If the cutout is covered up, the disk is "write-protected" and such commands cannot be carried out. As the optical device works in the visible light range, using a piece of transparent sticky tape (as you might do on a microdrive cartridge) to cover the cutout is not effective; the disk will not be protected unless the covering tape is opaque. Should you run out of the supplied write-protect tabs, you can use ordinary black plastic insulating tape. The disk pouch is not very thick, so heed the advice not to use an ordinary pen or pencil when writing on labels; use a felt-tip pen, which will not apply so much pressure to the magnetic disk.

If there is no identification on the pouch, you won't be able, visually, to tell whether a disk is single-, double- or high-density. You may still not be sure after a format operation, with some computers. The reason for this is that the disk capacity is not fixed thing. To some extent, it is dependent upon the drive and the format parameters used. A DD disk can be formatted in a HD drive to give well above



its official capacity, but don't waste your time trying this, as disks formatted this way cannot be trusted. You are virtually certain to get errors when using disks at above their rated capacity. This comment has to be qualified because of the point already made about being able to format 720KB 3½in disks to 1.44MB, and use them fairly successfully.

The 3½in disk looks much the same whether DD or HD, but there is one point of difference. The DD disk has one, small square cutout, equipped with a small slider. This is for write-protection and is again an optical device. When you cannot see through the hole — when the slider is pushed away from the edge of the disk — the disk can be written to. Pushing the slider fully outwards uncovers the hole and makes it impossible for a drive to write information onto it. This is philosophically the reverse of the situation with the 5¼in type, where the disk is write-protected when the hole is covered. On the HD disk casing, there is a matching square hole at the other corner, with no slider. The drive looks for this hole and, if it finds it, knows the disk is the 1.44MB type, with the tracks much closer together than the normal 720KB type.

The magnetic disk is covered by a sliding metal panel, which is moved out of the way by a mechanism inside the drive; so long as you don't deliberately slide the panel out of the way, and pour your coffee inside, there should be relatively little chance of damage occurring. Wear does occur on all types of floppy disk because the read/write heads actually sit in contact with the magnetic disk, in contrast to hard disks where the heads float above the surface. You can write on the label with whatever you like, without fear of affecting the disk inside. The central hub also is much less vulnerable than that in the 5¼in disk.

The smarter folk may ask, "Will I be able to get 1.44MB from a DD disk if I file a square hole in the other corner?", and the answer would appear to be "yes". That's not to say it's a sensible thing to do, though. It would be a delicate and time-consuming job, and there would be a distinct risk of plastic dust getting into the casing and causing errors. The other point is that there may be some difference between the magnetic media used for the different disk densities. Obviously, manufacturers are not going to be in a hurry to reveal what differences — if any — there are. A couple of ideas have been suggested to me; that DD disks are merely those HD disks which did not reach the quality control standards to be sold as HD, and that HD drives actually use a different recording bias voltage level to DD drives. Clearly, the magnetic disks would have to be separated out into DD and HD groups before putting into the casings, if the idea of DD disks being pure "second-grade" HD ones is true.

In practice, I have found that all of the 3½in DD disks I have formatted to 1.44MB (more than a hundred) can be used in *my* drive on the PC/AT, but I understand that this procedure would not work on most HD drives. The reason is that my drive does not appear to have an optical device for checking what the disk density is, and relies on a separate switch to do this. I came close to returning it to the shop at first, because it didn't seem to be a HD drive at all, until I discovered the jumpers which set the density. The answer to this problem was to put an external switch on the PC and use that whenever the disk type changed.

There have been a few errors on my PC when copying files off DD disks which have been used at HD; I mean "a few", but, nevertheless, floppy disk errors have been virtually non-existent on my QL and PC otherwise, and the implication appears to be that DD disks cannot be relied upon 100% when used HD.

In case anyone is still under an illusion at this point, let me repeat that the maximum capacity a QL drive can handle is 720KB. Neither interface nor drive (of

whatever type) is made to handle HD disks. The reason for commenting about the HD type is that some QL users also have (or have the use of) other types of computer, which do use HD disks. Miracle Systems say the potential trouble, in the way of user enquiries/complaints, the incompatibility that would be introduced, and the limited profitability, keep them from bringing out an HD drive for the QL. Another reason is the availability of the hard disk unit, which makes moving up to 1.44MB floppies a bit pointless; it wouldn't cost a lot less than going straight to hard disk.

Price is a major factor with most buyers, so what are the disadvantages of buying the cheapest disks on offer? It might be wise to split users into two categories here. If you are a "business user", for whom even a single disk error could spell disaster, you might be safest buying only branded (expensive) disks. This is especially true if you are not technically-minded, and are unable to deal with computer system problems. You should, however, be able to avoid major disasters by keeping backup copies (more than one, if the data is important). The home user usually does not have to worry so much, and does not need to bother so much about where disks come from. My own experience over the last five years is that unbranded disks are quite ok, although I prefer to have some sort of assurance from the suppliers that they trust their goods.

The first sign of good faith is accepting credit cards for purchases. This applies to whatever is supplied; simply don't trust most suppliers who want cash or cheque only. There are some cases where the supplier has little option; offering credit card facilities costs money, and requires a level of business sufficient to meet the credit companies' ideas of suitability for being granted the facility. If a supplier cannot manage to offer credit card facilities, it is perhaps better that they do not sell disks, because a lot of money has to be "put up front" to buy enough disks to get low prices. An advertised statement such as "100% guaranteed disks, money back if not satisfied" may be meaningless in some cases but, if no such assurance is given, one has even more reason to be nervous.

Until the All Formats Computer Fairs this year, I had believed that a sensible price to pay for 3½in disks, in lots of 10 or 25, was 60-90 pence each. Having now seen prices of as low as 50 pence, I have to revise my target. For 60-90p, you can get disks which are "guaranteed", and have some sort of a brand on them (eg "Benchmark"); pay less than that and you can expect neither guarantee nor brand name. At the moment, the one lot of 25 DD disks I bought for 38p each have had little use, although all of them formatted, first time, to 1.44MB and they don't look or sound in any way inferior to any other disks I've got. Time will tell how good an investment they were. Disks with the brands of well-known manufacturers (Sony, Maxell, TDK, Kao, 3M, Dysan etc) generally cost about £1.10 each upwards in lots of 10. Typical prices for unbranded 5¼in 96 tpi (QL) disks are from 40p up, and well-known brands can be had for about 65p upwards. 48 tpi (PC only) disks are not noticeably cheaper when unbranded, but branded ones start at around 60p each. Always make sure you've added post and packing and VAT into your calculations; the prices quoted above all include them.

As a general comment, increasing the "usability" of your system by adding disk drives is, initially, an alternative to adding more memory. To some extent, they each enable programs like Quill to be used with less pain, because the slowness of microdrive operation is made less of an influence on overall system speed. Marginally, I feel disk drives are the first priority, since they get over the major problem of the unreliability of microdrives. That is, you have greater data security. This may matter little to you, for example if you play games which require plenty of memory but don't need any serious amount of storage. In any event, the user who buys drives is likely to want extra memory also, and vice-versa, so what we are really looking at here is combined memory expansion and disk interface units.

A lot of words could be written about the numerous interfaces that have been sold for the QL, but virtually all the types are classifiable as "history" now and only those sold by Miracle are likely to be of interest to most users. However, there is a fairly strong market in secondhand interfaces, so something needs to be said about the older types.

The first supplier I can remember offering an interface and floppy drives for the QL was Quest. The units were large, and expensive. They are not a good buy now, unless *very* cheap; even then, only the user with some knowledge of computer systems should venture to buy them. You need software on cartridge or disk, with the drives, as (presumably) the driver routines are loaded from disk rather than being on rom in the interface. In this respect, these units are unique. Another type which needs to be treated with caution is the MCS (Micro Control Systems); you should check whether or not the "software patch" is being supplied (separately) with the interface. I accept Digital Precision's word (see the *Media Manager Special Edition* manual) that his is the only interface that has significant deficiencies in its software, at least in its earlier, unmodified form. Having said that, it depends largely on what software you use how much trouble you would have with the MCS interface.

The MicroPeripherals interface is likely to need the rom replacing by the QFLP one sold by Care Electronics.This is the interface which calls the floppy drive (FDK). The TR Delta interface seems basically sound, although I have had to investigate a few which went haywire, and had to have the rom chip upgraded; they appear to be able to blow chips in the QL as well, and I wouldn't recommend them as a first choice.

Interfaces which should be ok are the Sandy SuperQ, CST, PCML and Silicon Express types. Odd things to look out for are a modified rom in the PCML which can make any memory expansion in it unavailable; the modification should only have been made if the interface was fitted to a QL which already had an internal memory expansion fitted. Run your (or another standard) QL with the PCML fitted, and check that the full memory is displayed on the start-up screen. Early SuperQ interfaces had some trouble related to the PAL chips on them, and I think later versions did not have these chips. The CST was the first good buy, and remained the fastest one for a long time. Most interfaces could be supplied with some extra memory on them.

No article dealing with disk interfaces should be without some mention of Medic, if only because that brand once looked to be the answer to everyone's QL prayers. In principle, the Medic products were very good. In practice, they had deficiencies. Because of the relatively large quantity of Medic stock which got onto the market one way or another, and the obvious potential of the devices, various enthusiasts have worked to overcome the weaknesses. A properly-modified Medic interface is worth having. How to detect what is good and what is not, I can't say, and a thorough test in your own QL is suggested.

Coming back to the present, the Miracle Trump Cards are well worth buying and are, in fact, the only choice you have when buying new. They can have 0, 256KB or 768KB extra memory fitted on them. Older Trump Cards were slower than the current ones, and supported only two disk drives as against the present four. They also consumed more power, creating increased potential for lockups. When buying an old, secondhand Trump Card, try to run it for a few hours *in your house,* to see if lockups occur. Some old ones will have been modified to reduce lockups, by the fitting of eight resistors alongside a chip at the top right of the board (viewed as fitted to the QL).

When talking about interfaces, the subject of toolkits comes up automatically. As far as I know, there have been no interfaces without some form of extensions to SuperBasic incorporated. The standard is *Toolkit II* and almost all makes of interface will have some of the commands from this built-in, but only to the level of *Toolkit I* in most cases. The Trump Card definitely has Toolkit II built-in. With the others, it depends to some extent whether or not an upgraded rom has been fitted; just which extra commands were installed in which roms, only Tony Tebby is likely to be able to say.

ARCHMORE

Bryan Davies relays a program for adding fields to Archive

Archive has its merits, and perhaps the main one is the procedure language which is intrinsic to it. To the casual user, however, it can be a very frustrating program. If you have ever created a database file and the decided that you needed to add fields to it, you will have discovered that an apprently simple operation can turn out to be a pain.

The advice offered in the *QL User Guide* is to create a new database, with the revised fields and names, and Append the existing file to that.The Guide does give an example of how to write the procedure for doing this. No doubt I'm not the only one who had problems in both understanding and using this procedure at first. There may be several other ways of performing the operation, and one has been sent in to us by a reader in Belgium. The SuperBasic listing for the routine "ARCHmore", given here, was provided by Jean-Pierre Maquet of Liege. He is very interested in the use of Archive, and says he has written some procedures as large as 30KB. He also has as many as ten files open at the same time, apparently without problem.

The listing is in the original, unexpurgated French, so be very careful to type-in *exactly* what is printed. Some variables differ from each other by only one letter and can easily be mixed up.

Now to proceed: type in the SuperBasic listing and save it under the name ARCHmore. Run Archive, Open the file that needs the field structure changing, then Export it. *Close the file.* Archive automatically adds the extension _EXP to whatever name you give the Export-ed file.Leave Archive and run ARCHmore. The on-screen prompts tell you what to do from there on. You have to supply the _EXP file name, the name required for the converted file, the number of fields to be added, and the names of those fields.

When entering the name of the source file — that is, you Export-ed database — give it as "device_file", without the _EXP extention; for example, "flip_test". The destination file name — that is, the Export file with the new field names added — should be typed-in as "device_newfile", again without the extension. As the program adds the _EXP extension to the converted file also, you have to specify a different file name for it, to avoid the program trying to overwrite what it is reading. The converted file should then be Imported into Archive, and saved in the normal way as a _DBF file. to play safe, rename your original _DBF file and keep it on file until you are satisfied that the converted file is as you require it.

```
1 REMark **** ARCHMORE ****
2 REMark ADD NEW FIELDS TO AN ARCHIVE _EXP FILE
3 REMark written by J.F. MAQUET - LIEGE - BELGIUM
4 WINDOW 430,200,41,0:BORDER 6,5,2
5 CLS:CSIZE 2,1:start$=DATE$
6 PRINT "!! File names"!"should include"!"drive name;"
  !"default"!"extension"!"is"!"<_EXP>"
7 CSIZE 0,0:PRINT:CSIZE 1,1
8 INPUT "Name of source file.....? =";sourc$
9 IF sourc$(LEN(sourc$)-3)<>"_" OR LEN(sourc$)=8:sourc$=sourc$&"_EXP":END IF
10 INPUT "Name of file to be created...? =";dest$
11 IF dest$(LEN(dest$)-3)<>"_" OR LEN(dest$)=8:dest$=dest$&"_EXP":END IF
12 INPUT "Number of fields to be added ? =";nrub:CLS
13 adlign$="":rubr$=""
14 FOR n=1 TO nrub
15 INPUT "Name of "&n&": new field..... ? =";rub$
16 adlig$=','
17 IF rub$(LEN(rub$))="$"
18 adlig$=',""':Remark single quote/comma/2 double quotes/single quote
19 END IF
20 rub$=',"'&rub$&'"'
21 adlign$=adlign$&adlig$
22 rubr$=rubr$&rub$
23 END FOR n
24 OPEN#4,sourc$
25 OPEN_NEW#5,dest$
26 entre
27 PRINT#5;ligne$&rubr$
28 REPeat ajoute
29 IF EOF(#4):EXIT ajoute:END IF
30 entre
31 LET ligne$=ligne$&adlign$
32 PRINT#5;ligne$
33 END REPeat ajoute
34 CLOSE#4:CLOSE#5
35 BEEP 5000,5:CLS
36 PRINT\\\"                    PROCESS IS COMPLETE"\\
37 fin$=DATE$:PRINT"Time now:  "&fin$(13 TO)&" * started at: "&start$(13 TO)
38 DEFine PROCedure entre
39 ligne$=""
40 REPeat parlettre
41 a$=INKEY$(#4,-1)
42 IF CODE(a$)<26:EXIT parlettre:END IF
43 ligne$=ligne$&a$
44 IF EOF(#4):EXIT parlettre:END IF
45 END REPeat parlettre
46 END DEFine
```

Patience

The creation of the revised file is not fast, so be patient. Using the Create and Append process in Archive is not fast either. Adding two fields to a sample file of 9KB took 21 minutes. The program prints the start and finish times on the screen, when the conversion has been completed. The new field names are added after the last of the existing ones. One point to remember is that any Screen you have specially created to use in place of Archive's own default presentation of fields and data will not have been altered by the conversion process, and you will have to use Sedit to change it.

TOOLKIT TUTORIAL

Part 2

Mike Lloyd responds to queries arising from the Toolkit tutorial article.

The article on Tony Tebby's Super Toolkit II prompted a number of letters asking for closer examination of TK2 facilities. To begin this series, B. H. Jones of Easton-in-Gordano wrote before the article was published asking for clarification of some of the TK2 direct file access features.

Most difficulties encountered by Super Toolkit II users seem to stem from an incomplete understanding of the manual. To keep publishing costs to a minimum the manual concentrates on the particulars of each keyword and assumes that readers understand the background. Tebby's approach has denied people who are less than fully aware of the QL programming environment the use of many TK2 keywords. This occasional series is designed to fill the gaps in the TK2 manual and to encourage readers to make full use of the excellent facilities.

TK2 supplements SuperBasic standard file handling facilities to allow direct access to the contents of a file. A file is a stream of bytes recorded on a disc or Microdrive. A SuperBasic string variable can therefore be used as an analogy for a file. Characters in a string can be read or over-written individually. The TK2 commands BGET and BPUT provide the same facilities for direct file access.

To demonstrate these abilities it is assumed that a string called A$ is equal to "Sinclair QL World" and that a routine is needed which will convert the word "World" to capitals. These lines fit the bill:

```
100 FOR char = 13 TO 17
110 IF A$(char) > 96 THEN
120  A$(char) =
   CHR$(CODE(A$(char)) - 32)
130 END IF
140 END FOR char
```

The loop control variable char acts as a pointer, beginning at an arbitrary point in the string and then highlighting each subsequent character in turn. Whenever a file is opened there is an implicit pointer just like char which controls were file activity will take place. TK2 provides the means to control the pointer from within a program.

Assume that a file called f1p1_file contains the same text as A$. A channel to the file must be opened before it can be manipulated:

```
100 OPEN #3, f1pi_file
```

The word "World" must now be over-written by "WORLD". BGET and BPUT are both procedures and both handle byte values, integers ranging from 0 to 255, and so ASCII codes must be used to specify characters. Values outside this range generate errors:

```
110 FOR char = 13 TO 17
120 BGET#3\ char, CharVal
130 IF CharVal > 96 THEN
140 BPUT#3\ char, CharVal - 32
150 END IF
160 END FOR char
170 CLOSE #3
```

This routine is longer than the equivalent process for a string but the lines which do the work – 120 to 150 – are much simpler than before. Note that the backslash used in the BGET and BPUT statements is a special parameter separator.

Automatically

It is not always necessary to specify a character position or to give a value to be read or written. BPUT and BGET move the file pointer automatically so that the example process can be re-written as:

```
100 OPEN#3, f1pi_file
110 B$ = "WORLD"
120 BPUT#3\13
130 FOR char = 1 TO 5
140 PBUT#3, CODE(B$(char))
150 END FOR char
160 CLOSE#3
```

The file pointer is moved to the start of the word "World" by line 120. It then increments itself with each BPUT statement in the FOR..NEXT loop. Direct file access commands always over-write rather than displace existing characters in the file; there is no equivalent of the "insert mode" found in word processors.

BPUT can write any number of values in a single statement, each being separated by a comma. The final variation of the routine moves the file pointer and writes the ASCII values of "WORLD" with a single command:

```
100 OPEN#3, f1pi_file
110 BPUT#3\13, 87, 79, 82, 76, 68
120 CLOSE#3
```

Screens and printers can be written to as if they were files, provided that the file pointer position is not changed. This line prints the printable ASCII set in Window#1:

```
100 FOR char = 32 TO 191: BPUT char
```

To wherever the file pointer has been moved, the standard INPUT and PRINT commands can be used to communicate with the file. To add "Magazine" to "Sinclair QL World" the file pointer can be placed at the end of the file and the text added with a PRINT command. The FLEN function, also part of TK2, returns the file length:

```
100 OPEN#3, f1pi_file
110 BGET#3\ FLEN(#3)
120 PRINT#3, "Magazine"
130 CLOSE#3
```

The other direct access file commands, PUT, GET, TRUNCATE and FLUSH, will be considered in a later article.

P+R:O=G<S

If you have a program worthy of consideration, send it to 'The Progs', Sinclair QL World, Panini House, 116-120 Goswell Road, London EC1V 7QD. We pay for everything published at the usual rates.

CHEMISTRY

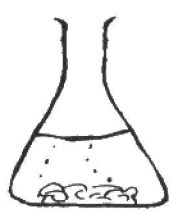
by Ian Thompson

The listing for "Chemistry" is fairly lengthy (about 10 KB), although it is straightforward to type in. The purpose of the program is to display data about chemical elements — the Periodic Table, relative atomic masses of (all) elements, elements and their symbols. Speedier operation will be obtained by compiling the SuperBasic code, using *Supercharge*, *Turbo* etc. The SB code has been "cleaned" using the Better Basic program, so should compile without difficulty. There is no problem with the speed of display using the SB version, however. Watch out for spelling mistakes when typing-in the element names.

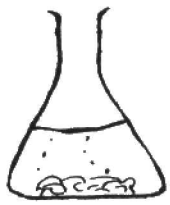
The on-screen menu is self-explanatory. The colour display of the Periodic Table suffers slightly from using stipples, which rarely allow clear characters to be read easily. It would be simple to alter the colour choices for some screen areas, but the problem is that more than the four basic QL colours are needed; however, some experimentation with the stipples should effect improvement. The Table displays Metals, Metaloids, Non-Metals, Noble Gases, and the Lanthanides and Actinides Series.

```
"CHEMISTRY"
100 REMark Chemistry
110 REMark by Ian Thompson
120 init
130 REPeat loop
140 menu
150 END REPeat loop
160 STOP
170 REMark ********************
180 DEFine PROCedure init
190 WINDOW 512,240,0,0:PAPER 0:INK 7:CLS
200 WINDOW#0,512,10,0,240
210 CSIZE 2,1:PRINT 'Please Wait - Initialising Arrays':CSIZE 0,0
220 DIM an$(106,3),name$(106,13),symbol$(106,2),ram$(106,7)
230 RESTORE 1900
240 FOR n=1 TO 106
250 READ a$,n$,s$,r$
260 an$(n)=a$
270 name$(n)=n$
280 symbol$(n)=s$
290 ram$(n)=r$
300 END FOR n
310 CLS
320 END DEFine
330 REMark ********************
340 DEFine PROCedure periodic_table
350 PAPER 0:INK 7:CLS
360 RESTORE 2960
370 FOR n=2 TO 14 STEP 2
380 READ letter$
390 AT n,2:PRINT letter$
400 AT n+1,4:PRINT (n/2)
410 END FOR n
420 mtable
430 PAPER 0:INK 7
440 AT 0,30:PRINT 'COLOUR KEY'
450 ckey
```

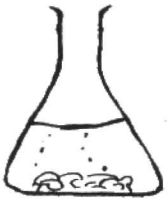
```
460 AT 6,28:PRINT 'G R O U P S'
470 AT 1,7:PRINT '1A'
480 AT 3,11:PRINT '2A'
490 AT 1,75:PRINT '0'
500 RESTORE 2970
510 FOR n=4 TO 40 STEP 4
520 READ num$
530 AT 7,n+11:PRINT num$&"B"
540 END FOR n
550 RESTORE 2980
560 FOR n=4 TO 20 STEP 4
570 READ num$
580 AT 3,n+51:PRINT num$&"A"
590 END FOR n
600 AT 16,27:PRINT 'Not Named'
610 AT 22,22:PRINT '< T H E  P E R I O D I C  T A B L E >'
620 AT 18,3:PRINT "Lanthanides"
630 AT 20,5:PRINT "Actinides"
640 PAPER 7,4,2:INK 0
650 AT 12,14:PRINT '(57-':AT 13,14:PRINT ' 71)'
660 PAPER 4,0:INK 7
670 AT 14,14:PRINT '(89-':AT 15,14:PRINT '103)'
680 PAPER 0:INK 7
700 END DEFine
710 REMark ********************
720 DEFine PROCedure mtable
730 PAPER 4:INK 0
740 AT 2,6:PRINT ' H  ':AT 3,6:PRINT ' 1  '
750 PAPER 4,2:INK 7:AT 2,74:PRINT ' He ':AT 3,74:PRINT ' 2
760 PAPER 0:INK 0
770 psym 6,4,2,7,7,0,0,3
780 psym 54,4,1,2,2,0,7,5
790 psym 58,4,4,4,4,0,0,6
800 psym 74,4,1,4,2,3,7,10
810 psym 6,6,2,7,7,0,0,11
820 psym 54,6,1,7,7,0,0,13
830 psym 58,6,1,2,2,0,7,14
840 psym 62,6,3,4,4,0,0,15
850 psym 74,6,1,4,2,3,7,18
860 psym 6,8,13,7,7,0,0,19
870 psym 58,8,2,2,2,0,7,33
880 psym 66,8,2,4,4,0,0,35
890 psym 74,8,1,4,2,3,7,36
900 psym 6,10,14,7,7,0,0,37
910 psym 62,10,2,2,2,0,7,51
920 psym 70,10,1,4,4,0,0,53
930 psym 74,10,1,4,2,3,7,54
940 psym 6,12,2,7,7,0,0,55
950 psym 18,12,12,7,7,0,0,72
960 psym 14,17,15,7,4,2,0,57
970 psym 66,12,2,2,2,0,7,84
980 psym 74,12,1,4,2,3,7,86
990 psym 6,14,2,7,7,0,0,87
1000 psym 18,14,3,7,7,0,0,104
1010 psym 14,19,15,4,0,2,7,89
1020 END DEFine
1030 REMark ******************
1040 DEFine PROCedure psym(row,col,noe,p1,p2,st,in,san)
1050 count=0
1060 PAPER p1,p2,st:INK in
1070 count=count+1
1080 AT col,row
1090 PRINT ' '&symbol$(san)&' '
1100 AT col+1,row
1110 PRINT ' '&an$(san)
1120 IF count<noe THEN : row=row+4:san=san+1:GO TO 1070
```



```
1130 END DEFine
1140 REMark ********************
1150 DEFine PROCedure ckey
1160 PAPER 7:AT 2,22:PRINT '   ':PAPER 0
1170 AT 2,25:PRINT "Metals"
1180 PAPER 2:AT 3,22:PRINT '   ':PAPER 0
1190 AT 3,25:PRINT 'Metaloids'
1200 PAPER 4:AT 4,22:PRINT '   ':PAPER 0
1210 AT 4,25:PRINT 'Non-Metals'
1220 PAPER 4,2:AT 2,38:PRINT '   ':PAPER 0
1230 AT 2,41:PRINT "Noble Gases"
1240 PAPER 7,4,2:AT 3,38:PRINT '   ':PAPER 0
1250 AT 3,41:PRINT 'Lan. Series'
1260 PAPER 4,0,2:AT 4,38:PRINT '   ':PAPER 0
1270 AT 4,41:PRINT 'Act. Series'
1280 END DEFine
1290 REMark ********************
1300 DEFine PROCedure elements
1310 PAPER 0:INK 7:CLS
1320 AT 0,30:PRINT '< E L E M E N T S >'
1330 FOR n=1 TO 22
1340 AT 1+n,2:PRINT symbol$(n)&' '&name$(n)
1350 END FOR n
1360 FOR n=23 TO 44
1370 AT n-21,16:PRINT symbol$(n)&' '&name$(n)
1380 END FOR n
1390 FOR n=45 TO 66
1400 AT n-43,31:PRINT symbol$(n)&' '&name$(n)
1410 END FOR n
1420 FOR n=67 TO 88
1430 AT n-65,48:PRINT symbol$(n)&' '&name$(n)
1440 END FOR n
1450 FOR n=89 TO 106
1460 AT n-87,62:PRINT symbol$(n)&' '&name$(n)
1470 END FOR n
1480 PRINT#0,'Press any key to return to the menu.':PAUSE
1490 END DEFine
1500 REMark ********************
1510 DEFine PROCedure RAM
1520 PAPER 0:INK 7:CLS
1530 AT 0,16:PRINT '< R E L A T I V E   A T O M I C   M A S S E S >'
1540 FOR n=1 TO 22
1550 AT 1+n,2:PRINT symbol$(n):AT 1+n,5:PRINT ram$(n)
1560 END FOR n
1570 FOR n=23 TO 44
1580 AT n-21,18:PRINT symbol$(n):AT n-21,21:PRINT ram$(n)
1590 END FOR n
1600 FOR n=45 TO 66
1610 AT n-43,34:PRINT symbol$(n):AT n-43,37:PRINT ram$(n)
1620 END FOR n
1630 FOR n=67 TO 88
1640 AT n-65,50:PRINT symbol$(n):AT n-65,53:PRINT ram$(n)
1650 END FOR n
1660 FOR n=89 TO 106
1670 AT n-87,66:PRINT symbol$(n):AT n-87,69:PRINT ram$(n)
1680 END FOR n
1690 PRINT #0,'Press any key to return to the menu.':PAUSE
1700 END DEFine
1710 REMark ********************
1720 DEFine PROCedure menu
1730 PAPER 0:INK 7:CLS#0:CLS
1740 CSIZE 2,1:UNDER 1
1750 AT 0,15:PRINT ' CHEMISTRY
1760 CSIZE 1,1:UNDER 0
1770 AT 2,16:PRINT 'This program will display :-
1780 PRINT '1) The Periodic Table'
```



```
1790 PRINT '2) The Relative Atomic Masses of all the elements'
1800 PRINT '3) The Elements and their symbols'
1810 PRINT:AT 7,16:PRINT 'Choose your option (1,2 or 3)'
1820 CSIZE 0,0
1830 opt$=INKEY$
1840 IF opt$='' OR opt$<'0' OR opt$>'3' THEN : GO TO 1740
1850 IF opt$='1' THEN :periodic_table
1860 IF opt$='2' THEN :RAM
1870 IF opt$='3' THEN :elements
1880 END DEFine
1890 REMark **********************
1900 DATA '1  ','Hydrogen','H ','  1.008'
1910 DATA '2  ','Helium','He','  4.003'
1920 DATA '3  ','Lithium','Li','  6.939'
1930 DATA '4  ','Beryllium','Be','  9.012'
1940 DATA '5  ','Boron','B ',' 10.811'
1950 DATA '6  ','Carbon','C ',' 12.011'
1960 DATA '7  ','Nitrogen','N ',' 14.007'
1970 DATA '8  ','Oxygen','O ',' 15.999'
1980 DATA '9  ','Fluorine','F ',' 18.998'
1990 DATA '10 ','Neon','Ne',' 20.183'
2000 DATA '11 ','Sodium','Na',' 22.990'
2010 DATA '12 ','Magnesium','Mg',' 24.312'
2020 DATA '13 ','Aluminium','Al',' 26.981'
2030 DATA '14 ','Silicon','Si',' 28.086'
2040 DATA '15 ','Phosphorus','P ',' 30.974'
2050 DATA '16 ','Sulphur','S ',' 32.064'
2060 DATA '17 ','Chlorine','Cl',' 35.453'
2070 DATA '18 ','Argon','Ar',' 39.948'
2080 DATA '19 ','Potassium','K ',' 39.102'
2090 DATA '20 ','Calcium','Ca',' 40.080'
2100 DATA '21 ','Scandium','Sc',' 44.956'
2110 DATA '22 ','Titanium','Ti',' 47.900'
2120 DATA '23 ','Vanadium','V ',' 50.942'
2130 DATA '24 ','Chromium','Cr',' 51.996'
2140 DATA '25 ','Manganese','Mn',' 54.938'
2150 DATA '26 ','Iron','Fe',' 55.847'
2160 DATA '27 ','Cobalt','Co',' 58.933'
2170 DATA '28 ','Nickel','Ni',' 58.710'
2180 DATA '29 ','Copper','Cu',' 63.540'
2190 DATA '30 ','Zinc','Zn',' 65.370'
2200 DATA '31 ','Gallium','Ga',' 69.720'
2210 DATA '32 ','Germanium','Ge',' 72.590'
2220 DATA '33 ','Arsenic','As',' 74.992'
2230 DATA '34 ','Selenium','Se',' 78.960'
2240 DATA '35 ','Bromine','Br',' 79.909'
2250 DATA '36 ','Krypton','Kr',' 83.800'
2260 DATA '37 ','Rubidium','Rb',' 85.470'
2270 DATA '38 ','Strontium','Sr',' 87.620'
2280 DATA '39 ','Yttrium','Y ',' 88.905'
2290 DATA '40 ','Zirconium','Zr',' 91.220'
2300 DATA '41 ','Niobium','Nb',' 92.906'
2310 DATA '42 ','Molybdenum','Mo',' 95.940'
2320 DATA '43 ','Technetium','Tc',' 99.000'
2330 DATA '44 ','Ruthenium','Ru','101.070'
2340 DATA '45 ','Rhodium','Rh','102.905'
2350 DATA '46 ','Palladium','Pd','106.400'
2360 DATA '47 ','Silver','Ag','107.870'
2370 DATA '48 ','Cadmium','Cd','112.400'
2380 DATA '49 ','Indium','In','114.820'
2390 DATA '50 ','Tin','Sn','118.690'
```



```
2400 DATA '51 ','Antimony','Sb','121.750'
2410 DATA '52 ','Tellurium','Te','127.600'
2420 DATA '53 ','Iodine','I ','126.904'
2430 DATA '54 ','Xenon','Xe','131.300'
2440 DATA '55 ','Caesium','Cs','132.905'
2450 DATA '56 ','Barium','Ba','137.340'
2460 DATA '57 ','Lanthanum','La','138.910'
2470 DATA '58 ','Cerium','Ce','140.120'
2480 DATA '59 ','Praseodymium','Pr','140.907'
2490 DATA '60 ','Neodymium','Nd','144.240'
2500 DATA '61 ','Promethium','Pm','147.000'
2510 DATA '62 ','Samarium','Sm','150.350'
2520 DATA '63 ','Europium','Eu','151.960'
2530 DATA '64 ','Gadolinium','Gd','157.250'
2540 DATA '65 ','Terbium','Tb','158.924'
2550 DATA '66 ','Dysprosium','Dy','162.500'
2560 DATA '67 ','Holmium','Ho','164.930'
2570 DATA '68 ','Erbium','Er','167.260'
2580 DATA '69 ','Thulium','Tm','168.934'
2590 DATA '70 ','Ytterbium','Yb','173.040'
2600 DATA '71 ','Lutetium','Lu','174.970'
2610 DATA '72 ','Hafnium','Hf','178.490'
2620 DATA '73 ','Tantalum','Ta','180.948'
2630 DATA '74 ','Tungsten','W ','183.850'
2640 DATA '75 ','Rhenium','Re','186.200'
2650 DATA '76 ','Osmium','Os','190.200'
2660 DATA '77 ','Iridium','Ir','192.200'
2670 DATA '78 ','Platinum','Pt','195.090'
2680 DATA '79 ','Gold','Au','196.967'
2690 DATA '80 ','Mercury','Hg','200.590'
2700 DATA '81 ','Thallium','Tl','204.370'
2710 DATA '82 ','Lead','Pb','207.190'
2720 DATA '83 ','Bismuth','Bi','208.980'
2730 DATA '84 ','Polonium','Po','209.000'
2740 DATA '85 ','Astatine','At','210.000'
2750 DATA '86 ','Radon','Rn','222.000'
2760 DATA '87 ','Francium','Fr','223.000'
2770 DATA '88 ','Radium','Ra','226.000'
2780 DATA '89 ','Actinium','Ac','227.000'
2790 DATA '90 ','Thorium','Th','232.038'
2800 DATA '91 ','Protactinium','Pa','231.000'
2810 DATA '92 ','Uranium','U ','238.030'
2820 DATA '93 ','Neptunium','Np','237.000'
2830 DATA '94 ','Plutonium','Pu','242.000'
2840 DATA '95 ','Americium','Am','243.000'
2850 DATA '96 ','Curium','Cm','247.000'
2860 DATA '97 ','Berkelium','Bk','247.001'
2870 DATA '98 ','Californium','Cf','251.000'
2880 DATA '99 ','Einsteinium','Es','254.000'
2890 DATA '100','Fermium','Fm','257.000'
2900 DATA '101','Mendelevium','Md','256.000'
2910 DATA '102','Nobelium','No','256.001'
2920 DATA '103','Lawrencium','Lw','257.001'
2930 DATA '104','Rutherfordium','Rf','259.000'
2940 DATA '105','Hahnium','Ha','260.000'
2950 DATA '106','(Not Named)','**','-------'
2960 DATA 'P','E','R','I','O','D','S'
2970 DATA '3','4','5','6','7','8','8','8','1','2'
2980 DATA '3','4','5','6','7'
```

DIR_TO_ARCHIVE_BAS

THE PROGS

by

T. McKnight

If you have a program worthy of consideration, send it to 'The Progs', Sinclair QL World, Greencoat House, Francis Street, London SW1P 1DG. We pay for everything published at the usual page rates.

DIR_TO_ARCHIVE_BAS was written as a way of sorting through microdrives or disks to locate files backed up several times on different drives, to achieve a sorted database file, through which the user can pick out files to delete or move around.

```
100 REMark init is at line 700 and all REMarks may be omitted if desired
110 REMark ************************************************************
120 REMark *                   DIR_TO_ARCHIVE_BAS                     *
130 REMark *                (c) T.Mcknight 7/05/90                    *
140 REMark ************************************************************
150 start:STOP
160 REMark ************************************************************
170 DEFine PROCedure                                            start
180 init
190 open_exp_file
200 REPeat data_input
210  prompt
220  REPeat ky
230   ky$=INKEY$(-1)
240   IF ky$=CHR$(10) THEN EXIT ky
250   IF ky$=CHR$(27)
260    no_more_data
270    EXIT data_input
280   END IF
290   IF ky$=' ' THEN STOP
300  END REPeat ky
310  clear_line_1:PRINT 'WAIT'
320  get_dir
330  read_tempdir
340  drive_nmbr=drive_nmbr+1
350 END REPeat data_input
360 END DEFine
370 REMark ************************************************************
380 DEFine PROCedure                                       open_exp_file
390 AT 1,5:PRINT 'CREATING EXPORT FILE'
400 DELETE media2$&'2_program_exp'
410 OPEN_NEW#4;media2$&'2_program_exp':REMark ensure drive is in 2
420 PRINT#4;'"rec_nmbr","drive_nmbr","title$"'\
430 FOR f=0 TO 5000:END FOR f
440 END DEFine
450 REMark ************************************************************
460 DEFine PROCedure                                            prompt
470 clear_line_1
480 PRINT 'Insert Cartridge ';drive_nmbr;
490 PRINT ' into drive 1 and press [ENTER] or [ESC] to stop'
500 END DEFine
510 REMark ************************************************************
520 DEFine PROCedure                                          get_dir
530 DELETE media2$&'2_tempdir':OPEN_NEW#5;media2$&'2_tempdir'
540 DIR#5;media1$
550 CLOSE#5
560 END DEFine
570 REMark ************************************************************
580 DEFine PROCedure                                       read_tempdir
590 OPEN_IN#5;media2$&'2_tempdir'
600 INPUT#5;dum1$:INPUT#5;dum2$
610 REPeat loop1
620   INPUT#5;title$
630   PRINT#4;rec_nmbr;',';drive_nmbr;',';'"';title$;'"'\
640   IF EOF(#5) THEN EXIT loop1
650   rec_nmbr=rec_nmbr+1
660 END REPeat loop1
670 CLOSE#5
680 END DEFine
690 REMark ************************************************************
700 DEFine PROCedure                                              init
710 MODE 4:WINDOW 512,256,0,0:CLS
720 rec_nmbr=1:drive_nmbr=1:media1$='mdv1_'
730 media2$='mdv':REMark CHANGE TO SUIT USER REQUIREMENTS i.e. FLP
740 prog$='DIR_TO_ARCHIVE_BAS':author$='(c) T.Mcknight 7th May 1990'
750 AT 0,5:PRINT prog$;' ';author$
760 END DEFine
770 REMark ************************************************************
780 DEFine PROCedure                                       no_more_data
790 PRINT#4;CHR$(26);
800 CLOSE#4
810 clear_line_1
820 PRINT 'COMPLETED Now reset and load ARCHIVE and import as per the notes'
830 END DEFine
840 REMark ************************************************************
850 DEFine PROCedure                                       clear_line_1
860 AT 1,0:PRINT TO 80:AT 1,5
870 END DEFine
880 REMark ************************************************************
890 DEFine PROCedure                                           save_me
900 init
910 AT 1,5:PRINT 'SAVING ';prog$
920 DELETE media2$&'2_'&prog$
930 SAVE media2$&'2_'&prog$, TO 1000
940 END DEFine
950 REMark FINISHED
```

After typing in the routine, Save it as DIR_TO_ARCHIVE-_BAS on mdv 2 by typing save_me and RUN. The program then asks the user to insert disk or microdrive 1 (it helps if you number all your microdrives or disks) and press ENTER or ESC to finish.

Repeat the process for all your media and then press ESC to close the file. For microdrives, this will take a long time because the drives are painfully slow but it is worth the wait. The user now has a PROGRAM_EXP file on mdv2 which is in the form which *Archive* is able to read.

Reset the machine (if you have the unexpanded QL) and load Archive. After loading, type IMPORT "PROGRAM" AS "PROGS"

ORDER title$;a,drive_nmbr;a
DUMP
CLOSE "PROGS"

This will dump the entire file to the printer in alphabetical order ready for the user to peruse. The user will now have the file "PROGS_DBF" on mdv2 which can be manipulated using Archive in any way the user likes. The structure of the file is:

rec_nmbr
drive_nmbr
title$

QUIT from the program, reset and load in your favourite file management utility such as the excellent FTIDY_BAS by Howard. J Clase, or use Superbasic to move files around.

The program works on any QL expansion and can be modified to suit your own requirements for different media by changing media1$ and media2$ in line 730 to 'mdv', 'ram' (if using a toolkit insert FORMAT RAM7_10 or equivalent ahead of media1$) or 'FLP' etc.